Jón Atli Benediktsson
Josef Kittler
Fabio Roli (Eds.)

LNCS 5519

Multiple Classifier Systems

8th International Workshop, MCS 2009
Reykjavik, Iceland, June 2009
Proceedings

MCS
2009

Springer

Jón Atli Benediktsson Josef Kittler
Fabio Roli (Eds.)

Multiple Classifier Systems

8th International Workshop, MCS 2009
Reykjavik, Iceland, June 10-12, 2009
Proceedings

Springer

Volume Editors

Jón Atli Benediktsson
University of Iceland
Faculty of Electrical and Computer Engineering
107 Reykjavik, Iceland
E-mail: benedikt@hi.is

Josef Kittler
University of Surrey
Centre for Vision, Speech and Signal Processing
Guildford, Surrey GU2 7XH, United Kingdom
E-mail: J.Kittler@eim.surrey.ac.uk

Fabio Roli
University of Cagliari
Department of Electrical and Electronic Engineering
09123, Cagliari, Italy
E-mail: roli@diee.unica.it

Library of Congress Control Number: Applied for

CR Subject Classification (1998): I.5, I.4, I.2.10, I.2, F.1

LNCS Sublibrary: SL 6 – Image Processing, Computer Vision, Pattern Recognition, and Graphics

ISSN 0302-9743
ISBN-10 3-642-02325-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-02325-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12689149 06/3180 5 4 3 2 1 0

Preface

These proceedings are a record of the Multiple Classifier Systems Workshop, MCS 2009, held at the University of Iceland, Reykjavik, Iceland in June 2009. Being the eighth in a well-established series of meetings providing an international forum for the discussion of issues in multiple classifier system design, the workshop achieved its objective of bringing together researchers from diverse communities (neural networks, pattern recognition, machine learning and statistics) concerned with this research topic.

From more than 70 submissions, the Program Committee selected 54 papers to create an interesting scientific program. The special focus of MCS 2009 was on the application of multiple classifier systems in remote sensing. This particular application uses multiple classifiers for raw data fusion, feature level fusion and decision level fusion. In addition to the excellent regular submission in the technical program, outstanding contributions were made by invited speakers Melba Crawford from Purdue University and Zhi-Hua Zhou of Nanjing University. Papers of these talks are included in these workshop proceedings. With the workshop's application focus being on remote sensing, Prof. Crawford's expertise in the use of multiple classification systems in this context made the discussions on this topic at MCS 2009 particularly fruitful.

As usual, the workshop would not have been possible without the help of many individuals and organizations. First of all, our thanks go to the members of the MCS 2009 Program Committee, whose expertise and dedication helped us create an interesting event that marks the progress made in this field over the last two years and aspires to chart its future research. The help of Shirley Hankers from the University of Surrey, who administered the submitted papers review, and of Björn Waske from the University of Iceland, who compiled the camera-ready manuscripts into a well-structured volume deserve a particular mention. The co-sponsorship of the event by the International Association for Pattern Recognition and its Technical Committee TC1: Statistical Techniques in Pattern Recognition, the IEEE Geoscience and Remote Sensing Society, the IEEE Iceland Section, the University of Iceland, the University of Cagliari and the University of Surrey is greatly appreciated and gratefully acknowledged.

June 2009

Jón Atli Benediktsson
Josef Kittler
Fabio Roli

Organization

The MCS 2009 was organized by the Faculty of Electrical and Computer Engineering of the University of Iceland in assossiation with the Center for Vision, Speech and Signal Processing of the University of Surrey, UK and the Department of Electrical and Electronic Engineering of the University of Cagliari, Italy.

Sponsoring Institutions

– International Association for Pattern Recognition and its Technical Committee TC1: Statistical Techniques in Pattern Recognition
– IEEE Geoscience and Remote Sensing Society, the IEEE Iceland Section
– University of Iceland
– University of Cagliari
– University of Surrey

Table of Contents

The Bias Variance Trade-Off in Bootstrapped Error Correcting Output Code Ensembles

Raymond S. Smith and Terry Windeatt

Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, Surrey GU2 7XH, UK
{Raymond.Smith,T.Windeatt}@surrey.ac.uk

Abstract. By performing experiments on publicly available multi-class datasets we examine the effect of bootstrapping on the bias/variance behaviour of error-correcting output code ensembles. We present evidence to show that the general trend is for bootstrapping to reduce variance but to slightly increase bias error. This generally leads to an improvement in the lowest attainable ensemble error, however this is not always the case and bootstrapping appears to be most useful on datasets where the non-bootstrapped ensemble classifier is prone to overfitting.

1 Introduction

When considering the errors made by statistical pattern classifiers it is useful to group them under three headings. Firstly there is the unavoidable error, known as *Bayes error*, which is caused by noise in the process that generates the patterns. A second source of error is *variance*; this is caused by the sensitivity of a learning algorithm to the chance details of a particular training set and causes slightly different training sets to produce classifiers that give different predictions for some patterns. Thirdly there are errors caused by *bias* in a learning algorithm; here the problem is that the classifier is unable, for whatever reason, to adequately model the class decision boundaries in the pattern feature space. When training a classifier there is often a tradeoff between bias and variance [10] so that a high value of one implies a low value of the other.

A successful approach to constructing multi-class classifiers has proved to be that of error-correcting output code (ECOC) ensembles [7,11]. In this approach the multi-class problem is decomposed into a series of 2-class problems, or dichotomies, and a separate base classifier trained to solve each one. These 2-class problems are constructed by repeatedly partitioning the set of target classes into pairs of super-classes so that, given a large enough number of such partitions, each target class can be uniquely represented as the intersection of the super-classes to which it belongs. The classification of a previously unseen pattern is then performed by applying each of the base classifiers so as to make decisions about the super-class membership of the pattern. Redundancy can be introduced into the scheme by using more than the minimum number of base classifiers and this allows errors made by some of the classifiers to be corrected by the ensemble as a whole. It has been shown [12,14] that ECOC reduces both bias and variance when compared with a single multi-class classifier.

A generally desirable property of multiple classifier systems (MCS), of which ECOC is an example, is that there should be *diversity* among the individual classifiers in the ensemble [4,17]. By this is meant that the errors made by component classifiers should, as far as possible, be uncorrelated so that the error correcting properties of the ensemble can have maximum effect. One way of achieving this is to apply bootstrapping to the training set so that each base classifier is trained on a unique bootstrap replicate. These are created from the original training set by repeated sampling with replacement. This creates a training set which has, on average, 63% of the patterns in the original set but with some patterns repeated to form a training set of the same size.

When bootstrapping is used in a majority voting ensemble of identical classifiers it leads to the technique of *bagging* [2]. This is known to reduce variance at the cost of increased bias [3,8], particularly when using an unstable classifier such as MLP. The situation with ECOC bootstrapping is somewhat analogous to a bagged ensemble; the difference, however, is that in the latter case each classifier is trained to solve an identical problem, whereas the ECOC base classifiers are trained to solve different sub-problems.

One of the advantages of the ECOC approach is that it makes it possible to perform multi-class classification by using base classifier algorithms that are more suited to solving 2-class problems. Examples include support vector machines (SVMs) [5] and multi-layer perceptron (MLP) neural networks [1]. In this paper we investigate experimentally three types of base classifier, namely single hidden layer MLPs, Gaussian kernel SVMs and polynomial kernel SVMs. Each of these base classifier types can be regarded as being controlled by two main parameters which respectively control the *capacity* and the *training strength* of the learning algorithm. The term *capacity* [5] refers to the ability of an algorithm to learn a training set with low or zero training error. By *training strength* we mean the amount of effort that is put into training the classifier to learn the details of a given training set. Intuitively, high capacity tends to imply a low bias and high training strength tends to imply high variance. For a given dataset and learning algorithm therefore, there is often a tradeoff between the values of these two parameters.

2 Kohavi-Wolpert Definition of Bias and Variance

The statistical concepts of bias, variance and noise originally emerged from regression theory. In this context they can be defined in such a way that the squared loss can be expressed as the sum of noise, bias^2 and variance. The goal of generalising these concepts to classification problems, using a 0-1 or other loss function, has proved elusive and several alternative definitions have been proposed (see [12] for a summary). In fact it is shown in [12] that, for a general loss function, these concepts cannot be defined in such a way as to possess all desirable properties simultaneously. For example the different sources of error may not be additive, or it may be possible for variance to take negative values.

In this study we adopt the Kohavi-Wolpert definitions [13]. Let X be a random variable representing input patterns and Y a random variable representing the target classes. Consider a learning algorithm $\mathcal{L}$ which, given a training set T, produces a classification function $\mathcal{L}(T)$ which maps X to Y. Then the Kohavi-Wolpert definitions of bias, variance and total error are given by the following equations:

$$bias_x^2 = \frac{1}{2}\sum_{y \in Y}\left[\hat{P}_{Y,X}(Y = y|X = x) - \hat{P}_T(\mathcal{L}(T)(x) = y)\right]^2 - D_x \tag{1}$$

$$variance_x = \frac{1}{2}\left[1 - \sum_{y \in Y}\hat{P}_T(\mathcal{L}(T)(x) = y)^2\right] + D_x \tag{2}$$

$$D_x = \frac{1}{2}\sum_{y \in Y}\hat{P}_T(\mathcal{L}(T)(x) = y)\left[1 - \hat{P}_T(\mathcal{L}(T)(x) = y)\right] / (N_T - 1) \tag{3}$$

$$error_x = \hat{P}_{Y,X}(\mathcal{L}(T)(x) \neq Y|X = x) = bias^2 + variance \tag{4}$$

Here $\hat{P}_{Y,X}(Y = y|X = x)$ is the empirical probability that the actual class of pattern x is y; in practice this takes the value 1 for a particular value of y and 0 for all others. $\hat{P}_T(\mathcal{L}(T)(x) = y)$ is the empirical probability, taken over a collection of N_T training sets, that the learning algorithm produces a classifier that assigns pattern x to target class y. D_x is a de-biasing term which ensures that the estimates of $bias_x^2$ and $variance_x$ are reliable for small values of N_T. The ability to apply this correction is one of the advantages of the Kohavi-Wolpert definitions; for example in [13] it is shown to lead to stable results using just 10 sample training sets.

Another advantage of the above definitions is that they give an additive decomposition of error. A major problem, however, is that there is no separate allowance for Bayes error. The rationale for this is that, on realistic datasets, this component of error cannot be estimated because the sampling is rarely dense enough to allow the probabilities of different classes to be estimated at a fixed value of x (a method for overcoming this problem has, however, been proposed in [12]). In effect, the Bayes error component is absorbed into the bias^2 term, thus giving a value which is biased too high. In this study, however, we are interested only in changes to bias and variance as the base classifier parameters are varied, and so this issue does not affect the conclusions of the paper.

3 ECOC Base Classifier Parameters

As noted in section 1, we wish to characterise the base classifier parameters as those which control the capacity of the classifier and those which control the training strength. In the case of single hidden layer MLPs, a natural choice is to take the number of hidden nodes and the number of training epochs respectively.

For SVM base classifiers note that the objective function to be minimised during training [5] has the form $\|\mathbf{w}\|^2 + C\sum_i \xi_i$ where $\mathbf{w}$ is the weight vector to be computed, C is a cost parameter and ξ_i are slack variables. The value of C controls the tradeoff between exactly fitting the training data (by driving the ξ_i towards zero) and maximising the margin (by driving $\|\mathbf{w}\|$ towards zero). It follows that C fulfils the role of the training strength parameter, with high values leading to the training set being modelled more precisely.

The choice of capacity parameter for SVMs depends on the kernel function being used. The Gaussian kernel has the form $exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{y}\|^2\right)$ where σ controls the diameter of the sphere of influence around each support vector. For this kernel $1/\sigma^2$ is a suitable choice for the capacity parameter (the inverse is taken in order to ensure that capacity increases as the parameter value increases). Some pictorial examples of the effect of varying C and σ can be found in [16]. For the polynomial kernel function $(\mathbf{x}\cdot\mathbf{y}+1)^d$ the capacity is determined by the degree parameter d.

4 Experiments

In this section we present the results of performing classification experiments on 11 multi-class datasets obtained from the publicly available UCI repository [15]. The characteristics of these datasets in terms of size, number of classes and number of features are given in table 1.

Table 1. Experimental datasets showing the number of patterns, classes, continuous and categorical features

Dataset	Num. Patterns	Num. Classes	Cont. Features	Cat. Features
dermatology	366	6	1	33
ecoli	336	8	5	2
glass	214	6	9	0
iris	150	3	4	0
segment	2310	7	19	0
soybean	683	19	0	35
thyroid	7200	3	6	15
vehicle	846	4	18	0
vowel	990	11	10	1
waveform	5000	3	40	0
yeast	1484	10	7	1

For each dataset, ECOC ensembles of size 200 were constructed using each of three base classifier types and a range of base classifier capacity and training strength parameters. Each such combination was repeated 10 times with different randomly chosen stratified training sets and different randomly generated ECOC coding matrices; for neural network base classifiers another source of random variation was the initial network weights. In each run the data was normalised to make the training set have zero mean and unit variance. The ECOC code matrices were constructed in such a way as to have balanced numbers of 1s and 0s in each column. Training sets were based on a 20/80 training/test set split. Each experiment was repeated with and without bootstrapping being applied to the construction of the individual base-classifier training sets. In total this led to 27,900 experimental runs being performed. For each unique combination of parameters and algorithms, the Kohavi-Wolpert bias^2, variance and total error were calculated in accordance with Eqns. 1 to 4 over the 10 randomised runs.

The base classifier types employed were single-hidden layer MLP neural networks using the Levenberg-Marquardt training algorithm, SVMs with Gaussian kernel and SVMs with polynomial kernel. The neural networks were constructed as a single hidden layer of perceptrons, with the number of nodes ranging from 2 to 16 and the number of training epochs from 2 to 1024. For Gaussian SVMs the width parameter σ was varied between 1 and 8, whilst for polynomial SVMs degrees of 1,2,3 and 4 were used. The cost parameter of SVMs was varied between 10^{-3} and 10^3. In all cases, apart from polynomial degrees, the base classifier parameters were varied in geometric progression.

For reference purposes a complete list of the lowest ensemble errors obtained in these experiments, for each base classifier type, is given in Table 2.

Table 2. The lowest percentage ensemble error values obtained using three types of ECOC base classifier. Error values are shown with the application of bootstrapping (BS) and without ($\overline{\text{BS}}$).

Dataset	Neural Network		Gaussian SVM		Polynomial SVM	
	BS	$\overline{\text{BS}}$	BS	$\overline{\text{BS}}$	BS	$\overline{\text{BS}}$
dermatology	**3.3**	4.8	**2.9**	**2.9**	**2.8**	3.2
ecoli	**16.8**	18.3	15.1	**15.0**	**15.7**	16.0
glass	**36.2**	36.8	35.5	**35.3**	**37.2**	38.0
iris	**4.8**	5.1	**5.0**	5.7	5.5	**5.3**
segment	**4.0**	**4.0**	**5.7**	**5.7**	**5.7**	6.1
soybean	9.7	**9.3**	8.4	**7.8**	8.3	**8.2**
thyroid	2.7	**2.6**	**2.7**	2.8	**2.9**	3.4
vehicle	**20.9**	22.1	**22.1**	22.2	**23.1**	23.5
vowel	23.2	**21.3**	21.3	**20.9**	26.4	**25.9**
waveform	**14.9**	16.7	**14.3**	14.4	**14.4**	14.5
yeast	41.9	**41.4**	41.2	**41.1**	**42.0**	**42.0**

4.1 Bias-Variance Tradeoff

Some representative examples of the bias-variance behaviour observed in these experiments are illustrated in Fig. 1. Here the effect is shown, both with and without bootstrapping, of increasing the training strength parameter for various datasets and base classifier types. For each graph the base classifier capacity parameter is fixed at the optimal value obtained on the test set for the given dataset.

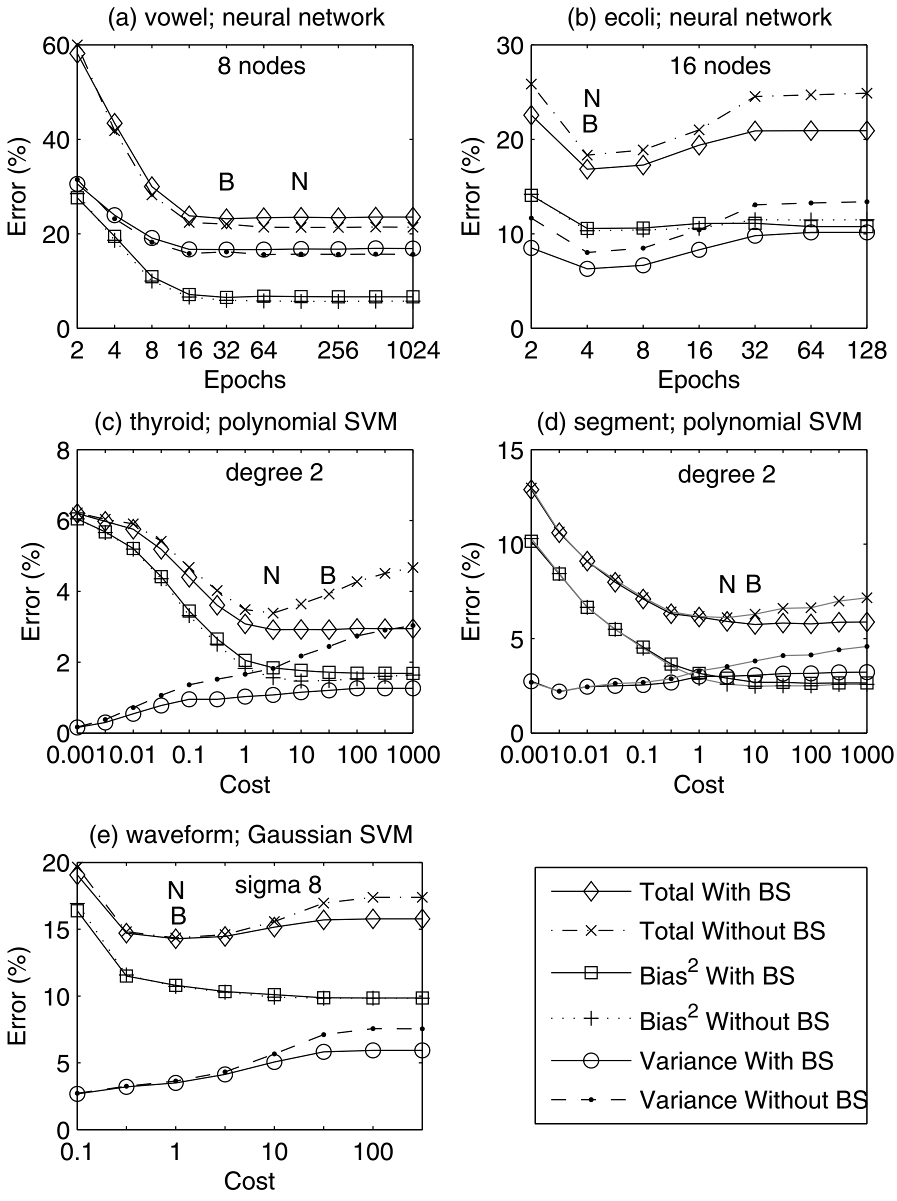


Fig. 1. Some example ensemble test-set bias, variance and total error curves as the ECOC base classifier training strength parameter is varied. These are shown with and without the application of bootstrapping during ensemble construction. 'B' and 'N' respectively mark the positions of minimum error with and without bootstrapping.

A number of observations can be made about Fig. 1. We discuss first examples (b) to (e) where it can be seen that, as expected, there is a general tendency for the bias^2 to decrease and the variance to increase as the training strength increases. The effect of bootstrapping on variance is to lessen this increase, particularly for higher values of the training strength parameter. It can be observed that there is a tendency for bootstrapping to slightly increase the bias^2 error, although the effect is usually small and the curves generally lie very close to each other. It is noteworthy that the bootstrapped variance and total error curves tend to level out at a lower value than the non-bootstrapped versions; this indicates that bootstrapping makes the ensemble more resistant to overfitting the data at high training strengths.

At some point there is an optimal tradeoff between bias and variance where the total error is minimised. The position of the optimum may vary depending on whether bootstrapping is used or not (e.g. (c) and (d)) or it may be the same in both cases (e.g. (b) and (e)). Whether bootstrapping reduces the total error depends on the values of bias^2 and variance at the optimal tradeoff points. In examples (b) to (d) the variance reduction induced by bootstrapping is sufficient to lead to a significant overall reduction in error despite any slight increase in bias. The benefit of bootstrapping tends to be lower, or even negative, however when the optimal bias/variance tradeoff occurs at low training strengths; this is because, as in case (e), the divergence between the variance curves is insufficient, at this point, to significantly impact the total error.

The behaviour observed on some datasets, for example case (a) of Fig. 1, can be different from that described above. Here the ECOC classifier does not exhibit a pronounced tendency to overfit the data at high training strengths (as in cases (b) to (e)) and, as a result, variance is not reduced by bootstrapping. In fact, in example (a) both the bias^2 and variance curves of the bootstrapped ensemble lie slightly above those of the non-bootstrapped version, so bootstrapping leads to an overall increase in total error.

4.2 Bootstrapping vs. Non-bootstrapping

In section 4.1 we examined the classification behaviour of ECOC ensembles under conditions of identical base classifier capacity and varying training strengths. In order to compare the performance of bootstrapped versus non-bootstrapped ensembles, however, it is necessary to look at them under optimal conditions and this may require the base classifier capacity, as well as training strength, to be different. The examples of Fig. 1 were chosen from cases where the optimal capacity was found to be the same for both types of ensemble, but this is not always the case. Due perhaps to its more stochastic behaviour, the neural network base classifier was found to be particularly prone to this phenomenon, with only 3 out of the 11 datasets requiring the same capacity parameter. For example on the yeast dataset this classifier was optimal at 4 nodes and 16 training epochs when bootstrapping was used but 8 nodes and 8 epochs when not.

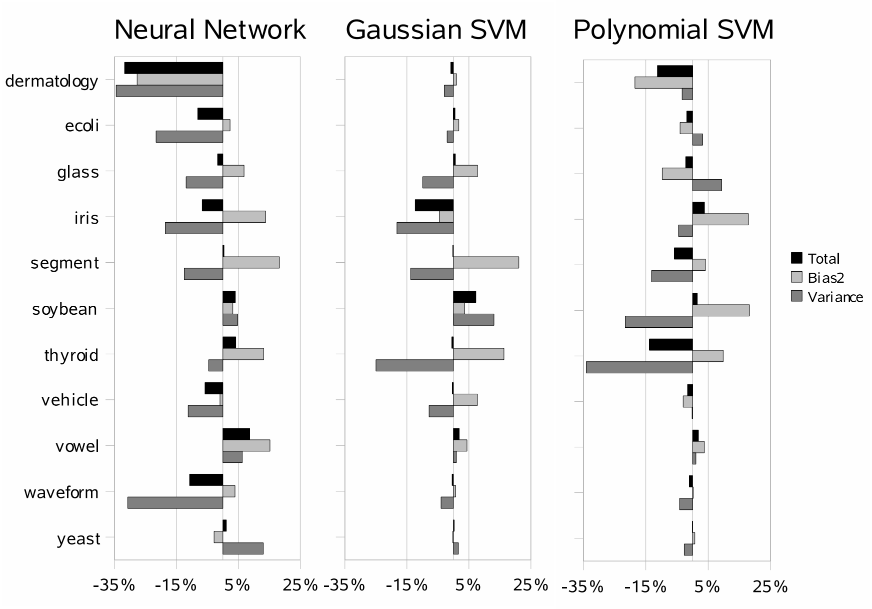


Fig. 2. The effect of bootstrapping on ECOC for three types of base classifier. Figures show the relative percentage change in the lowest attainable ensemble Kohavi-Wolpert test error components that result from bootstrapping. Negative values imply that bootstrapping leads to a reduction.

Table 3. The average, over 11 datasets, of the effect of applying bootstrapping to ECOC. Figures show the mean percentage relative change in the lowest attainable ensemble Kohavi-Wolpert test error measures. Negative values imply that bootstrapping leads to a reduction.

Base Classifier	Total	Bias^2	Variance
Neural Network	-4.22	4.12	-11.05
Gaussian SVM	-0.36	5.44	-6.18
Polynomial SVM	-2.78	1.77	-6.38

Fig. 2 shows the relative percentage change[1] in test-set bias^2, variance and total error which resulted when the bootstrapped ensemble was compared with the non-bootstrapped version at their respective points of minimum total error. It can be seen from this that the general pattern is for bootstrapping to reduce variance but to increase bias and that this leads to a net reduction in total error. This pattern of behaviour is confirmed by Table 3 which shows the relative percentage changes averaged over the 11 datasets. There are, however, deviations from this pattern for individual datasets. For example dermatology, when using

[1] By relative percentage change we mean the value $100\,(v - v_{BS})\,/v$ where v and v_{BS} are measured without and with bootstrapping respectively.

the neural network or polynomial SVM base classifiers, leads to the bias^2 at the point of bootstrapped minimum total error being significantly less than that obtained when bootstrapping is not applied.

5 Discussion and Conclusions

The main contribution of this paper to our understanding of ensemble classifiers is to shed light on how the bootstrapping of ECOC ensembles affects performance, not just in terms of overall classification error, but also how that error breaks down into its bias and variance components. Evidence has been presented to show that bootstrapping generally tends to lessen the impact of variance when compared with non-bootstrapped ensembles. This tends to be particularly noticeable at high values of the training strength parameter, leading to a reduced tendency to overtrain. The relative reduction in variance is, however, often achieved at the expense of a slight increase in the bias^2 component - a pattern of behaviour that is reminiscent of that observed in bagged ensembles [8].

Whilst the net effect of bootstrapping is usually to reduce the overall error that can be attained at optimal base classifier parameter settings, this is not universally the case and bootstrapping appears to be most useful on datasets for which the non-bootstrapped ensemble is prone to overfitting. This is to be expected since the latter type of dataset implies that variance error plays a more prominent role in determining the ensemble error.

Future work will be directed towards characterising more precisely the relationship between the properties of the dataset and the effect of ECOC bootstrapping. For example, when the available dataset is small, as with iris, it is likely that further reducing the base classifier training data by bootstrapping may lead to the introduction of bias. This cannot be the complete explanation, however, as increases in bias can also be observed on larger datasets such as thyroid and segment. Further investigation is required and it is hoped that this will lead to a theory that predicts when bootstrapping is advantageous.

Acknowledgements

This work was supported by EPSRC grant E061664/1. Thanks are also due to the providers of the prtools [9] and libsvm [6] software.

References

1. Bishop, M.C.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
2. Breiman, L.: Bagging Predictors. Machine Learning 24(2), 123–140 (1994)
3. Breiman, L.: Arcing Classifiers. Annals of Statistics 26(3), 801–849 (1998)
4. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity Creation Methods: A Survey and Categorisation. Journal of Information Fusion 6(1) (2005)

5. Burges, C.J.C.: A Tutorial on Support Vector Machines for Pattern Recognition. Knowledge Discovery and Data Mining 2(2) (1998)
6. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
7. Dietterich, T.G., Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes. Journal of Artificial Intelligence Research 2, 263–286 (1995)
8. Dietterich, T.G., Kong, E.B.: Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical Report, Dept. of Computer Science, Oregon State University (1995)
9. Duin, R.P.W., Juszczak, P., Paclik, P., Pekalska, E., de Ridder, D., Tax, D.M.J., Verzakov, S.: PRTools 4.1, A Matlab Toolbox for Pattern Recognition, Delft University of Technology (2007)
10. Geman, S., Bienenstock, E.: Neural networks and the bias / variance dilemma. Neural Computation 4, 1–58 (1992)
11. James, G.: Majority Vote Classifiers: Theory and Applications. Ph.D Dissertation, Stanford University (1998)
12. James, G.: Variance and Bias for General Loss Functions. Machine Learning 51(2), 115–135 (2003)
13. Kohavi, R., Wolpert, D.: Bias plus variance decomposition for zero-one loss functions. In: Proc. 13th International Conference on Machine Learning, pp. 275–283 (1996)
14. Kong, E.B., Dietterich, T.G.: Error-correcting output coding corrects bias and variance. In: Proc. 12th International Conference on Machine Learning, pp. 313–321 (1995)
15. Merz, C.J., Murphy, P.M.: UCI Repository of Machine Learning Databases (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
16. Valentini, G., Dietterich, T.G.: Bias-Variance Analysis of Support Vector Machines for the Development of SVM-Based Ensemble Methods. Journal of Machine Learning Research 5, 725–775 (2004)
17. Windeatt, T.: Accuracy/ Diversity and Ensemble Classifier Design. IEEE Trans. Neural Networks 17(4) (July 2006)

Recoding Error-Correcting Output Codes

Sergio Escalera, Oriol Pujol, and Petia Radeva

Computer Vision Center, Campus UAB, Edifici O, 08193, Bellaterra, Spain
Dept. Matemàtica Aplicada i Anàlisi, UB, Gran Via 585, 08007, Barcelona

Abstract. One of the most widely applied techniques to deal with multi-class categorization problems is the pairwise voting procedure. Recently, this classical approach has been embedded in the Error-Correcting Output Codes framework (ECOC). This framework is based on a coding step, where a set of binary problems are learnt and coded in a matrix, and a decoding step, where a new sample is tested and classified according to a comparison with the positions of the coded matrix. In this paper, we present a novel approach to redefine without retraining, in a problem-dependent way, the one-versus-one coding matrix so that the new coded information increases the generalization capability of the system. Moreover, the final classification can be tuned with the inclusion of a weighting matrix in the decoding step. The approach has been validated over several UCI Machine Learning repository data sets and two real multi-class problems: traffic sign and face categorization. The results show that performance improvements are obtained when comparing the new approach to one of the best ECOC designs (one-versus-one). Furthermore, the novel methodology obtains at least the same performance than the one-versus-one ECOC design.

1 Introduction

Recently, significant amount of robust binary classifiers have been proposed in the bibliography with very high performance, such as Support Vector Machines, Neural Networks, Adaboost [1], etc. However, the extension of many binary classifiers to the multi-class case, where N possible categories appear, is a hard task. In this sense, a common strategy consists of defining a set of binary problems, which are combined in a Multiple Classifier system.

Error-Correcting Output Codes (ECOC) were defined as a framework to combine binary problems in order to deal with the multi-class case [2]. This framework is based on two main steps. At the first step, named coding, a set of binary problems (dichotomizers) are defined based on the learning of different sub-partitions of classes by means of a base classifier. Then, each of the partitions is embedded as a column of a coding matrix M, which rows correspond to the codewords codifying each class. At the second step, named decoding, a new data sample that arrives to the system is tested, and a codeword formed as a result of the output of the binary problems is obtained. This test codeword is compared with each class codeword based on a given decoding measure, and a classification prediction is obtained for the new object. Unlike the voting procedure, the information provided by the ECOC dichotomizers are shared among classes in order to obtain a precise classification decision, being able to reduce either the variance as the bias produced by the learners [3].

When Dietterich et. al. defined the binary ECOC framework in [2], all positions from the coding matrix M belonged to the $\{+1, -1\}$ symbols. It makes all classes to be considered by each dichotomizer as a member of one of both possible partitions of classes that define each binary problem. In this case, the one-versus-all and dense random ECOC approaches were defined [2]. Afterwards, Allwein et. al. in [4] defined the ternary ECOC, where the positions of the coding matrix M can be either $+1$, -1 or 0, and the sparse random and one-versus-one (pairwise voting) designs could be defined in the ECOC framework. In this case, the zero symbol means that a given class is not considered in the learning process of a particular dichotomizer. The huge set of possible bi-partitions of classes from this ternary ECOC framework has recently suggested the use of problem-dependent designs as well as new decoding strategies[5][6][7][8][9].

Concerning the one-versus-one ECOC strategy, it codifies the splitting of each possible pair of classes as a dichotomizer, which results in $N(N-1)/2$ binary problems for an N-class problem. This number is usually larger in comparison with the linear tendency of the rest of ECOC designs. Although this suggests larger training times, the individual problems that we need to train on are significantly smaller, and if the training algorithm scales superlinearly with the training set size, it is actually possible to save time. Moreover, the problems to be learnt are usually easier, since the classes have less overlapping. For all these reasons, the one-versus-one ECOC design tends to obtain better results than the rest of ECOC designs in real multi-class problems[5][7].

In this paper, we focus on the one-versus-one coding matrix design. Our goal is to look for a better coding of the matrix without retraining the classifiers involved. Training data are used in a problem-dependent way for updating the zero positions to +1 or -1 symbols if a higher classification performance can be achieved. Observe the 4-classes problem shown in Fig. 1(a). A decision boundary of a non-linear classifier has been obtained in the learning process of the dichotomizer h_1 that splits classes c_1 and c_2. The point of this article is that without the necessity of retraining the classifier, the same decision boundary can be used to give a prediction hypothesis about class c_3. On the other hand, note that the use of this decision boundary to classify class c_4 may result in a random decision function. Using this information, we recode the classical problem-independent one-versus-one into a problem-dependent one-versus-one design extending the trained classifier on new classes for the binary classifier for which the dichotomizer is relevant. The design is possible thanks to a new weighting procedure that takes into account the performance of the dichotimizers at the decoding step [7]. Moreover, the approach requires almost the same training and testing computational complexity than the classical one (since retraining of classifiers is not required).

The paper is organized as follows: Section 2 describes the recoded problem-dependent one-versus-one approach. Section 3 evaluates the methodology over a set of UCI data sets and two real multi-class problems: traffic sign and faces categorization. Finally, section 4 concludes the paper.

2 Recoded One-versus-One ECOC

In this section, we present a problem-dependent redefinition of the classical one-versus-one ECOC design. The one-versus-one ECOC technique is defined in the ternary ECOC framework $M^{N \times M} \in \{-1, 0, +1\}$, being M a coding matrix of N rows (as the number of classes), M the number of columns (dichotomizers to be learnt, where $M = N(N-1)/2$ in the case of the one-versus-one design), $\{-1, +1\}$ symbols codify the class membership, and the zero symbol ignores a particular class for a given dichotomizer. Each column of the matrix M corresponds to the ith binary problem h_i, which splits a pair of classes using a given base classifier. Figure 1(b) codifies a coding matrix M for a 4-class problem. The white positions correspond to the symbol +1, the black positions to the symbol -1, and the grey positions to the zero symbol. Note that this design is independent from the problem-domain. Once the set of binary problems $h = \{h_1, .., h_M\}$ is learnt, a new test sample ρ that arrives to the system is tested applying the set h, and a test codeword $x^{1 \times M} \in \{-1, +1\}$ is obtained. Afterwards, a decoding function $d(x, y_j)$ is used to compare the test codeword x with each codeword y_j (jth row from M) codifying class c_j. Finally, the classification prediction corresponds to the class c_j which corresponding codeword y_j minimizes d.

In the one-versus-one ECOC design, only $2M$ from the NM possible positions are coded to $\{-1, +1\}$ symbols, which corresponds to a $(1 - 2/N) \cdot 100$ percentage of positions coded to zero. Note that the zero symbol does not give class membership information for its corresponding dichotomizer. Then, it could happen that if some of these positions coded to zero are re-coded to +1 or -1 without the need of re-training the dichotomizers, the final performance could be improved almost without increasing the training cost.

2.1 RECOC Coding

Given the training data $C = \{C_1, .., C_N\}$, where C_i is the data belonging to class c_i, and M the one-versus-one coding matrix, the set of dichotomizers $h = \{h_1, .., h_M\}$ is learnt applying a base classifier over the corresponding subsets of C, obtaining the classical one-versus-one ECOC design. In order to update the coding matrix in a problem-dependent way, for each position $M(i, j) = 0$, the corresponding data C_i, $i \in \{1, .., N\}$, $i \notin (k, l)$, where c_k and c_l are the classes considered by the jth dichotomizer, are tested using h_j under the hypothesis that their membership should be +1. Then, a classification accuracy β is obtained. If the magnitude of β or $(1 - \beta)$ is greater than a performance threshold $\alpha \in (0.5, 1]$, then that position of the coding matrix M is set (recoded) to +1 (or -1), respectively. Otherwise, the value of $M(i, j)$ is kept to zero.

Since we use the training data to modify the positions of M, the one-versus-one design mutates in a problem-dependent way. Moreover, since the modification of the positions of M does not require to retrain the set h, the computational cost of the coding process is not significantly increased. Table 1 shows the algorithm for training the Recoding ECOC (RECOC) design. The algorithm codifies the classical one-versus-one design at the same time that modifies the positions of M based on the input value of α. Note that in the algorithm, a matrix of weights W saving the accuracy values β is defined. This matrix will be used at the decoding process in order to weight the final classification.

Table 1. RECOC learning algorithm

```
Input: $\alpha$, $C = \{C_1, .., C_N\}$ // Accuracy value and multi-class data
Output: $M$, $W$, and set of dichotomizers $h = \{h_1, .., h_M\}$
$W^{N \times M} := 0$, $M^{N \times M} := 0$, $cont := 1$
for $i \in \{1, .., N-1\}$
    for $j \in \{i+1, .., N\}$
        Given a base classifier, learn dichotomizer $h_{cont}$ to split $(C_i, C_j)$
        // Update membership and accuracy
        $M(i, cont) := +1$, $W(i, cont) := h_{cont}(C_i, +1)$
        // Update membership and accuracy
        $M(j, cont) := -1$, $W(j, cont) := h_{cont}(C_j, -1)$
        for $k \in \{i, .., N\}$
            if $k \notin \{i, j\}$
                // Accuracy for class $c_k$ considered as class $c_i$ (label +1)
                $\beta := h_{cont}(C_k, +1)$
                // Consider the coding matrix position $k$ as +1
                if $\beta \geq \alpha$ then
                    // Update membership and accuracy
                    $M(k, cont) := +1$, $W(k, cont) := \beta$
                // Consider coding matrix position $k$ as -1
                elseif $1 - \beta \geq \alpha$ then
                    // Update membership and accuracy
                    $M(k, cont) := -1$, $W(k, cont) := 1 - \beta$
                endif
            endif
        endfor
        $cont := cont + 1$
    endfor
endfor
```

In order to obtain more precise classification results, we need to know which values of α are useful to increase the generalization capability of the system, since some values of α may result in wrong classification predictions. In order to look for the values of α, cross-validation is applied. For this task, the training data C is split into a training C^T and a validation C^V subsets, so that $C = C^T \cup C^V$. The use of a validation subset helps the system to increase generalization. Thus, for a set of values $\alpha = \{\alpha_1, .., \alpha_k\}$, algorithm 1 is called. However, the set h is only learnt once over C at the beginning. At each round, the set C^T is used to mutate the positions of M, and the validation set C^V will be used to test the performance of each M for a particular α. For this last task, a decoding procedure using the weighting matrix W is proposed next. This step is required to obtain a successful classification. Finally, the matrix M for which value of α maximizes the classification performance over C^V is selected.

Figure 1 shows an example of a training process for a 4-class problem. Figure 1(a) shows the non-linear decision boundaries that splits all possible

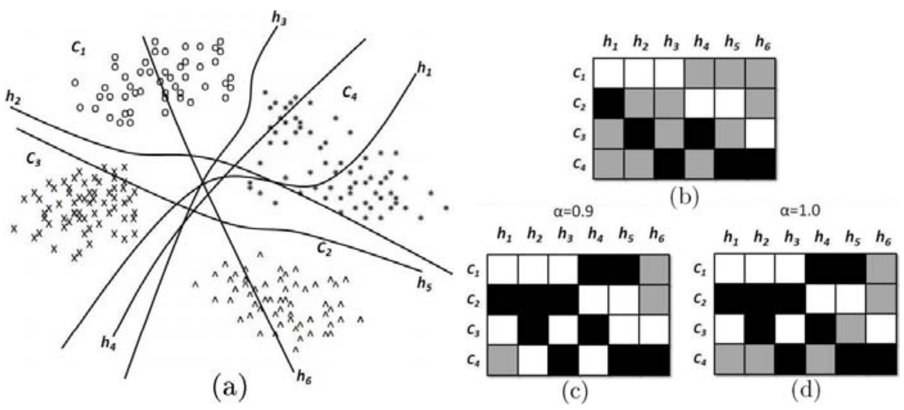


Fig. 1. ECOC codification for a 4-class problem: (a) Non-linear decision boundaries for the 4-class problem, (b) initial one-versus-one ECOC codification, (c) RECOC codification with $\alpha = 0.9$, and (d) RECOC codification with $\alpha = 1.0$

pairs of classes. Figure 1(b) shows the classical one-versus-one design. Figure 1(c) shows the problem-dependent coding matrix M for $\alpha = 0.9$. Note that several positions previously coded to zero are now set to +1 or -1 values since they achieve an accuracy upon 90% over the training data. Finally, Fig. 1(d) shows the same process for $\alpha = 1.0$. Now, less positions satisfy the performance restrictions. Note that if the testing of the validation data C^V does not take benefits from the values of α, then, the classical one-versus-one design is selected, and thus, in the worst case, the recoded problem-dependent approach attains the same performance than the classical approach.

2.2 RECOC Decoding

In [7], the authors show that to properly decode a ternary ECOC matrix two biases must be avoided at the decoding step. First, classical decoding strategies introduce a bias when comparing positions that contain the zero symbol, which do not give information about meta-class membership. On the other hand, the addition of the bias produced by the comparison with the zero symbol makes the codewords to take values from different ranges, which makes the measures among codewords non-comparable. In this sense, the authors present how to robustly decode sparse coding matrices where codewords may contain different number of positions coded to $\{-1, +1\}$ symbols. This is done by weighting the final decision so that it avoids the influence of the zero symbol at the same time that all classes codewords have the same probability of being predicted.

Due to the previous properties, we use a Loss-based decoding [4] weighted by the weighting matrix W computed at the RECOC coding step to decode the RECOC matrix M. The approach uses a Loss-function to penalize the miss-classifications produced by the set of dichotomizers h.

First, we normalize each row of the weighting matrix W obtained at the coding step so that M_W can be considered as a discrete probability density function $M_W(i,j) = \frac{W(i,j)}{\sum_{j=1}^{M} W(i,j)}$, $\forall i \in [1,...,N]$, $\forall j \in [1,...,M]$. Once we obtain the normalized weighting matrix M_W, we introduce it in a Loss-based decoding [4]. In this approach, the decoding estimation is obtained by means of a Loss-based model with a Loss-function $L(\theta)$ weighted by M_W, where $L(\theta) = -\theta$ and θ corresponds to $y_i^j \cdot h^j(\rho)$: $LW(\rho, i) = \sum_{j=1}^{n} M_W(i,j)L(y_i^j \cdot h_j(\rho))$. The final classification decision is done by the class c_i which corresponding codeword y_i that minimizes the LW function.

3 Results

In order to present the results, first, we discuss the data, methods, measurements, and experimental settings of the experiments.

• *Data*: The data used for the experiments consist of eleven multi-class data sets from the UCI Machine Learning Repository database [10]. We also categorize two real Computer Vision classification problems. First, we use the video sequences obtained from a Mobile Mapping System [11] to test the methods in a real traffic sign categorization problem consisting of 36 traffic sign classes. Second, 30 classes from the ARFaces [12] data set are classified using the present methodology.

• *Methods*: We compare the classical one-versus-one ECOC design with the RECOC strategy for three base classifiers: Gentle Adaboost [1], Linear Support Vector Machines [13], and Support Vector Machines with Radial Basis Function kernel ($RBF\ SVM$) [13]. In order to compare the methods at same conditions, we use a linear Loss-Weighted decoding in both one-versus-one and RECOC strategies.

• *Measurements*: To measure the performance of the different strategies, we apply stratified ten-fold cross-validation and test for confidence interval with a two-tailed t-test.

• *Experimental settings*: 50 decision stumps are considered for the Gentle Adaboost algorithm. The $RBF\ SVM$ classifier is tuned via cross-validation, where the σ and regularization parameters are tested from 0.05 increasing per 0.05 up to 1 and from one increasing per 5 up to 150, respectively. For the RECOC strategy cross-validation is applied, where α is tested from 0.7 increasing per 0.05 up to 1, and 10% of the training data are used as a validation subset.

3.1 UCI Classification

Table 2 shows the performance results of the one-versus-one ECOC and RECOC algorithms for the different ECOC base classifiers. For each UCI data set, the performance obtained by each method is shown. In the cases where RECOC improves the one-versus-one ECOC results, the selected values of α are shown. The number of wins, losses, and draws considering the ten experiments of the ten-fold cross-validation for each data set are also shown in the table. Note that

Table 2. UCI performances for the different ECOC base classifiers

Gentle Adaboost	one-versus-one	RECOC	α	Wins	Losses	Draws
Balance	**87.46**	**87.46**	-	0	0	10
Wine	**94.38**	**94.38**	-	0	0	10
Thyroid	**95.37**	**95.37**	-	0	0	10
Iris	**95.33**	**95.33**	-	0	0	10
Glass	63.10	**68.65**	0.95	10	0	0
Ecoli	81.29	**83.36**	0.75	8	2	0
Dermatology	91.76	**92.52**	0.85	5	0	5
Vowel	57.88	**62.73**	0.95	9	1	0
Vehicle	57.81	**63.57**	0.95	9	1	0
Yeast	55.46	**56.67**	0.95	5	2	3
Segmentation	**97.45**	**97.45**	-	0	0	10
Linear *SVM*	one-versus-one	RECOC	α	Wins	Losses	Draws
Balance	**91.64**	**91.64**	-	0	0	10
Wine	**95.55**	**95.55**	-	0	0	10
Thyroid	**96.71**	**96.71**	-	0	0	10
Iris	**98.67**	**98.67**	-	0	0	10
Glass	28.74	**37.58**	1.00	5	1	4
Ecoli	**74.63**	**74.63**	-	0	0	10
Dermatology	94.79	**95.07**	0.95	1	0	9
Vowel	63.33	**64.44**	0.95	8	2	0
Vehicle	**80.24**	**80.24**	-	0	0	10
Yeast	26.11	**37.81**	0.95	9	1	0
Segmentation	96.02	**96.32**	1.00	6	2	2
RBF SVM	one-versus-one	RECOC	α	Wins	Losses	Draws
Balance	97.25	**97.41**	0.95	1	0	9
Wine	61.31	**61.84**	1.00	1	0	9
Thyroid	**95.35**	**95.35**	-	0	0	10
Iris	**96.67**	**96.67**	-	0	0	10
Glass	**46.41**	**46.41**	-	0	0	10
Ecoli	**86.74**	**86.74**	-	0	0	10
Dermatology	88.80	**89.05**	0.85	3	0	7
Vowel	54.95	**55.76**	0.90	4	1	5
Vehicle	72.00	**72.12**	0.90	1	0	9
Yeast	**56.68**	**56.68**	-	0	0	10
Segmentation	95.14	**95.25**	0.90	2	0	8

in several data sets, RECOC obtains performance improvement for the three base classifiers. The table shows that the more classes there are, the more significant the results are. The highest performances are achieved for high values of α (about 0.90-0.95 in most cases). Note that in the worst case, RECOC becomes the one-versus-one ECOC designs, and it achieves the same performance. Moreover, looking at the wins and losses of each experiment, one can see that though in some case the performance improvements of RECOC are no significant, the number of wins of the ten-fold experiments are statistically significant.

Now, we compare the results obtained by the RECOC approach on the UCI data sets with the results obtained with the same strategy retraining classifiers. In Fig. 3 one can see the performance obtained by both classification strategies for the three different base classifiers. Note that there are no significant differences among the obtained performances. Moreover, the RECOC strategy obtains better performance in more cases than using the same coding matrix retraining classifiers, with far less computational complexity.

3.2 Traffic Sign Categorization

For this experiment, we use the video sequences obtained from the Mobile Mapping System [11] to test the classification methodology on a real traffic sign categorization problem. In this system, the position and orientation of the different traffic signs are measured with video cameras fixed on a moving vehicle.

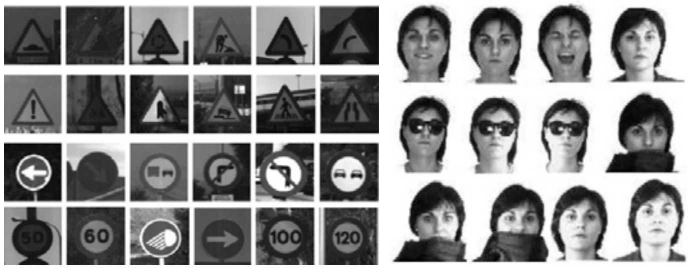


Fig. 2. (a) Traffic sign classes. (b) ARFaces data set classes. Examples from a category with neutral, smile, anger, scream expressions, wearing sun glasses, wearing sunglasses and left light on, wearing sun glasses and right light on, wearing scarf, wearing scarf and left light on, and wearing scarf and right light on.

Table 3. Traffic data set performances

Problem	one-versus-one	RECOC	α	Wins	Losses	Draws
Gentle Adaboost	88.70	88.95	0.95	3	1	6
Linear *SVM*	88.02	91.23	1.00	4	0	6
RBF SVM	97.44	97.85	0.95	1	0	9

From this system, a set of 36 circular and triangular traffic sign classes are obtained. Some categories from this data set are shown in Fig. 2(a). The data set contains a total of 3481 samples of size 32×32, filtered using the Weickert anisotropic filter, masked to exclude the background pixels, and equalized to prevent the effects of illumination changes. These feature vectors are then projected into a 100 feature vector by means of PCA. The classification results of the one-versus-one ECOC and RECOC strategies for the three base classifiers are shown in Table 3. In this experiment, for all base classifiers, the RECOC design obtains performance improvements for high values of α.

3.3 ARFaces Classification

The AR Face database [12] is composed of 26 face images from 126 different subjects (70 men and 56 women). The images have uniform white background. The database has two sets of images from each person, acquired in two different sessions, with the following structure: one sample of neutral frontal images, three samples with strong changes in the illumination, two samples with occlusions (scarf and glasses), four images combining occlusions and illumination changes, and three samples with gesture effects. One example of each type is plotted in Fig. 2(b). For this experiment, we selected all the samples from 30 different categories (persons).

The classification results of the one-versus-one ECOC and RECOC strategies for the three base classifiers are shown in Table 4. As in the previous experiments,

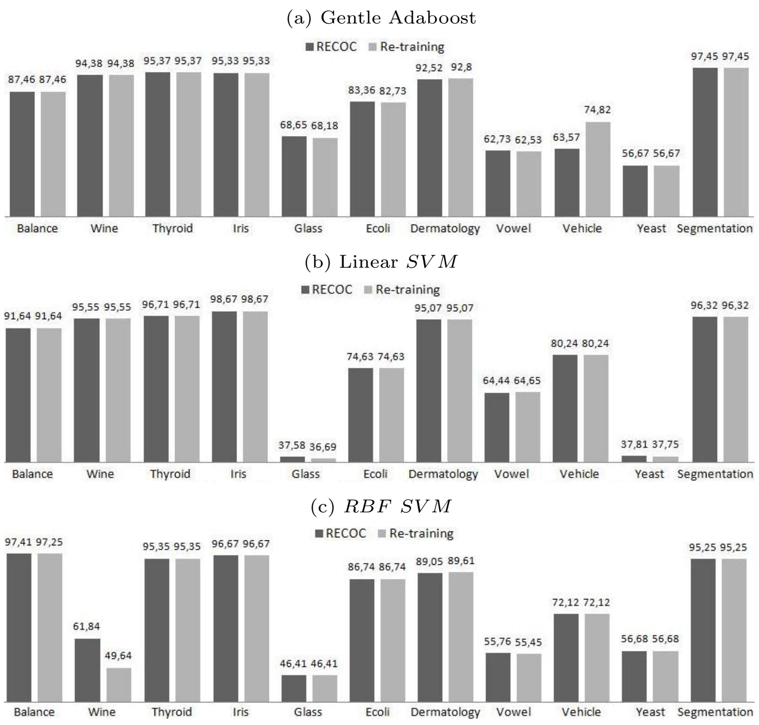


Fig. 3. UCI data sets performance using the recoded matrix with and without retraining

Table 4. ARFaces data set performances

Problem	one-versus-one	RECOC	α	Wins	Losses	Draws
Gentle Adaboost	65.50	70.06	0.95	6	1	3
Linear SVM	39.41	43.92	0.95	9	1	0
$RBFSVM$	88.33	88.75	0.95	2	0	8

all base classifiers obtain performance improvements using the RECOC strategy for high values of α ($\alpha = 0.95$).

3.4 Discussion

As a final conclusion of the results, we can state that performance improvements are obtained using the RECOC approach instead of the one-versus-one ECOC. Note that none of the RECOC experiments for any base classifier obtains inferior results to the one-versus-one performances.

Concerning the computational complexity of the strategy, the classifiers learnt at the coding step are not retrained during the RECOC recodification. Thus, though cross-validation of α should be applied to assure the better performance, the training cost is not significantly increased. On the other hand, the testing time remains the same than in the classical one-versus-one approach since all classifiers should be applied on the test sample. Moreover, we show that we obtain similar (even superior) results with the recoded RECOC matrix M than using the same procedure but retraining classifiers (that is, using the re-coded positions to re-train again the dichotomizers).

Finally, it is important to bring up that though the recoding strategy has been performed on the one-versus-one coding matrix, this strategy is directly applicable to any kind of ternary ECOC design where the symbol zero may appear.

4 Conclusion

In this paper, we presented a problem-dependent design of Error-Correcting Output Codes to deal with multi-class categorization problems. The method is based on redefining the classical one-versus-one ECOC design so that the generalization of the system is increased. For this task, the training data are analyzed using the previously learnt binary problems, and the coding matrix is recoded without the need of retraining classifiers. A weighting matrix is also included in order to weight the final classification and obtain more precise results. The experimental evaluation over several UCI Machine Learning repository data sets and two real multi-class problems: traffic sign and face categorization, show that significant performance improvements can be obtained. Moreover, our new methodology is guaranteed by design to achieve at least the one-versus-one performance.

Acknowledgments

This work has been supported in part by projects TIN2006-15308-C02, FIS PI061290, and CONSOLIDER-INGENIO CSD 2007-00018.

References

1. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. The annals of statistics 38, 337–374 (1998)
2. Dietterich, T., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2, 263–282 (1995)
3. Kong, E.B., Dietterich, T.G.: Error-correcting output coding corrects bias and variance. In: ICML, pp. 313–321 (1995)
4. Allwein, E., Schapire, R., Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers. JMLR 1, 113–141 (2002)
5. Escalera, S., Tax, D., Pujol, O., Radeva, P., Duin, R.: Subclass problem-dependent design of error-correcting output codes. Transactions in Pattern Analysis and Machine Intelligence 30, 1041–1054 (2008)
6. Pujol, O., Radeva, P., Vitrià, J.: Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes. In: PAMI, vol. 28, pp. 1001–1007 (2006)

7. Escalera, S., Pujol, O., Radeva, P.: On the decoding process in ternary error-correcting output codes. Transactions in Pattern Analysis and Machine I Intelligence (in press)
8. Escalera, S., Pujol, O., Radeva, P.: Boosted landmarks of contextual descriptors and Forest-ECOC: A novel framework to detect and classify objects in clutter scenes. Pattern Recognition Letters 28(13), 1759–1768 (2007)
9. Pujol, O., Escalera, S., Radeva, P.: An incremental node embedding technique for error correcting output codes. Pattern Recognition 41, 713–725 (2008)
10. Asuncion, A., Newman, D.: UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences (2007), http://mlearn.ics.uci.edu/MLRepository.html
11. Casacuberta, J., Miranda, J., Pla, M., Sanchez, S., Serra, A., Talaya, J.: On the accuracy and performance of the GeoMobil system. In: International Society for Photogrammetry and Remote Sensing (2004)
12. Martinez, A., Benavente, R.: The AR Face database. Computer Vision Center Technical Report #24 (1998)
13. Osu-svm-toolbox, http://svm.sourceforge.net

Comparison of Bagging and Boosting Algorithms on Sample and Feature Weighting

Satoshi Shirai*, Mineichi Kudo, and Atsuyoshi Nakamura

Division of Computer Science
Graduate School of Information Science and Technology
Hokkaido University, Sapporo 060-0814, Japan
{satoshi,mine,atsu}@main.ist.hokudai.ac.jp
http://prml.main.ist.hokudai.ac.jp

Abstract. We compared boosting with bagging in different strengths of learning algorithms for improving the performance of the set of classifiers to be fused. Our experimental results showed that boosting worked much with weak algorithms and bagging, especially feature-based bagging, worked much with strong algorithms. On the basis of these observations we developed a mixed fusion method in which randomly chosen features are used with a standard boosting method. As a result, it was confirmed that the proposed fusion method worked well regardless of learning algorithms.

1 Introduction

Recently, classifier fusion is gathering much attention in pattern recognition. Many studies have been seen in the series of *Multiple Classifier Systems* workshops. The main two streams for classifier fusion are *bagging* [1] and *boosting* [2]. Bagging utilizes several subsets of training samples, mainly by resampling, in order to stabilize classifiers constructed by those subsets. The random subspace method [3] can be also seen as one of bagging algorithms in the sense that it stabilizes the set of classifiers. It resamples subsets of features instead of samples. The boosting also utilizes randomly-resampled subsets of training samples, but in resampling it weights heavily the samples that were not correctly classified up to the current stage.

As a general understanding, bagging is effective when the *base learning algorithms*, producing each (component) classifier with a given training sample set, have a high ability to explain any set of training samples but are affected much by a little change of the samples. On the other hand, boosting is effective for the base learning algorithms with relatively low explanation ability but high robustness. These things are described in several ways. For examples, with the words of *bias* and *variance*, bagging is effective for algorithms with low bias and high variance and boosting is effective for algorithms with low variance and high bias. A great work in this way is seen in Friedman [4]. If we use the words of

* Corresponding author.

training error and *testing error*, bagging is effective for algorithms with a large difference between these two errors and boosting for algorithms with a small difference. This is important from a practical point of view, because if the difference is large, we cannot rely on the training error and thus we cannot choose the best one among some classifiers from their training errors. Then, model selection should be carried out, but unfortunately any model selection criterion tends to work poorly for up to a moderate size of training sample sets.

In this paper, we investigated to what degree the above discussion holds. That is, we evaluated several combinations of methodology (boosting and bagging) and learning algorithms (simple and complex). To enhance the discussions, we proposed an integrated algorithm in which samples are reweighted according to the failure of previous stages and features are reweighted according to the importance on the chosen subset of samples.

Our conclusions are 1) boosting is effective for simpler algorithms (in this paper a Plug-in Bayes linear classifier) whose difference between training and testing errors is small and bagging is effective for more complicated algorithms (in this paper a support vector machine) whose difference between training and testing errors is large, 2) Our boosting algorithm, in which random feature subsets are chosen so as to work better for the chosen random subset of samples, works almost always regardless of base algorithms, and 3) bagging using random feature subset selection works better than simple bagging on samples.

In the following, we will describe the previous studies on comparison between boosting and bagging, propose our algorithm as a merger of these two, and then show the experimental results to have derived the above conclusions.

2 Previous Studies

Simply speaking, boosting strengthens a set of weak classifiers to a stronger classifier, and bagging weakens a set of strong classifiers to a weaker classifier. The first insight for boosting is given in the way of reweighting on samples. It reweights heavily the samples misclassified so far to obtain a better (component) classifier on the resampled samples. As a result, the integrated classifier can perform beyond each of component classifiers even if each one is weak for a whole set of training samples. On the other hand, bagging produces a combined classifier, usually by majority vote on component classifiers. Since majority vote is equivalent to averaging, it can be expected that the variance between component classifiers is reduced. As a result, a more smooth decision boundary is obtained.

In bagging, there are several pieces of evidence showing that random feature selection (e.g., the random subspace method: `RSM`) works better than random sample selection used in the standard bagging [3,5]. This may be explained by the degree of *independence* of classifiers [5,6,7]. In general, for majority vote fusion, it is known that negatively correlated (component) classifiers much contribute to the improvement of the combined classifier, and even positively correlated classifiers have a chance for contribution. It is clearly true that randomly chosen small number of features increases the independency more than randomly chosen large number of samples. We cannot reduce the sample size to have a reliable classifier. Feature selection is rather preferable for this goal. Indeed, the random subspace method (shortly, `RSM`) is better than a regular bagging (shortly, `Bagging`) in many cases. This is also experimentally confirmed by our previous study [8].

Several observations have been reported in comparison between bagging and boosting [7,9,10,11]. Skurichina and Doin compared the performance of bagging, boosting and random subspace method using linear classifiers [7]. They said that, in the case of n samples and m features, 1) for $n/m < 1$, all methods are almost even; 2) for $1 \leq n/m \leq 3$, bagging and random subspace method is better than boosting; and 3) for $n/m > 3$, boosting is better than bagging and random subspace method. Skurichina and Kuncheva compared bagging with boosting by the diversity of component classifiers [9]. The divergence shows the degree of variety of outputs of component classifiers. In general, a higher value of divergence guarantees more the effectiveness of classifier fusion. In [9], it was shown that boosting outperforms bagging in both the nearest mean classifiers and a pseudo Fisher linear classifiers, and the diversity of boosting was larger than that of bagging. Quinlan also showed that boosting is better than bagging for C4.5 [10]. In [10], it is also reported that boosting could fail when the error concentrates on a certain class. In that case, boosting could be even harmful. Note that these studies use simpler learning algorithms such as linear classifiers and decision trees. For complicated learning algorithms, SVM was tested in [11]. Even for SVM, it was reported that boosting was better than bagging for hand-writing digit. These reports show that boosting is mostly better than bagging, but it is still not clear that in which cases this is true, especially in the relationship with learning algorithms.

Unfortunately, there is few of studies that compared bagging with boosting in different learning algorithms. It is important to know which combination is better than another. Thus, we will compare several combinations in the following.

3 Formulation of Fusion Algorithms

In this section, we provide the notation first and then describe several fusion algorithms.

3.1 Dataset Description

Let us assume that we are given a training sample set of n training samples in m-dimensional Euclidean space. We make ready two index sets: $I \subseteq I_0 = \{1, 2, \ldots, n\}$ for showing which samples of given n samples were used and $J \subseteq J_0 = \{1, 2, \ldots, m\}$ for showing which features of m given features were used. That is, $|I|$ samples in $|J|$ features are specified by I and J. In addition, for each i, we have the correct class label y_i. Then a training sample set, with a certain pair of I and J, chosen from the original sample set $X = \{x_{i,j} | i = 1, 2, \ldots, n, j = 1, 2, \ldots, m\}$ is denoted by $X_{I,J} = \{x_{i,j} \ | i \in I, j \in J\}$. Let $\phi(x; I, J) = A(X_{I,J})$

be a (component) classifier designed by a base learning algorithm A and a training set $X_{I,J}$. Here, $\phi(x; I, J)$ outputs an estimated class label for a given x. The final classifier is obtained from $\phi(x; I_1, J_1), \phi(x; I_2, J_2), \ldots, \phi(x; I_B, J_B)$ by majority vote (averaging) or by weighted averaging. That is, B component classifiers are combined into $\Phi(x) = \sum_{k=1}^{B} \beta_k \phi(x; I_k, J_k)$, where β_k is the weight of kth classifier.

The difference between boosting and bagging algorithms can be shown by the difference of I's and J's. Here, we allow duplications of elements in I but not in J. This is because some learning algorithms behave differently for a training sample set with and without duplication. In contrast to this, duplications of features are always harmful for designing classifiers.

`Bagging`: $I_1, I_2, \ldots, I_B$ ($|I_k| = n, k = 1, 2, \ldots, B$) are chosen from I_0 randomly and uniformly with replacement. $J_k = J_0 (k = 1, 2, \ldots, B)$. The combine weights are $\beta_k = 1/B (k = 1, 2, \ldots, B)$.
`Random Subspace Method`: $J_1, J_2, \ldots, J_B$ ($|J_k| \sim m/2, k = 1, 2, \ldots, B$) are chosen from J_0 randomly and uniformly without replacement.Typically, a half is chosen. $I_k = I_0 (k = 1, 2, \ldots, B)$. The combine weights are $\beta_k = 1/B (k = 1, 2, \ldots, B)$.
`Adaboost`: $I_1, I_2, \ldots, I_B$ ($|I_k| = n, k = 1, 2, \ldots, B$) are chosen from I_0 randomly according to sample weights $\{w_i\}$.The weights are updated according to previous misclassification. $J_k = J_0 (k = 1, 2, \ldots, B)$. The combine weights $\{\beta_k\}$ are taken so as to be higher if kth classifier has a smaller training error.

3.2 Base Learning Algorithm

Due to time limitation, we compared only two base learning algorithms. As a representative of simple classifiers, we used Plug-in Bayes linear classifier (shortly, `linear`). As a representative of strong classifiers, we used support vector machines (shortly, `SVM`). It is clear that `linear` is insufficient for the cases that the Bayes decision boundary is not linear. Instead, it gains robustness against the change of training samples. Therefore, *overfitting* (a large difference between the training and testing errors) can be avoided. On the other hand, `SVM` with a kernel (RBF is chosen in this paper) easily explains all the training samples with the correct class labels. Therefore, it tends to show overfitting for many problems, although its maximum margin criterion suppresses the degree low in theory.

4 `SF-Boost` Algorithm

As described in the preceding sections, it is expected that boosting works in the direction of increasing the complexity of component classifiers and bagging works in the reverse direction. The random feature selection also works in reducing the complexity, because a smaller set of features limits more the representation space of samples. However, it also can work for removing redundant features and, therefore, for bringing reliability on estimation of classifier parameters. As a result, it is worth checking the effectiveness of combination of boosting on samples and bagging (random selection) on features. We call it a *boosting with bidirectional weighting on Sample and Feature* (shortly, `SF-Boost`).

`SF-boost (Adaboost+RSM)`: $I_1, I_2, \ldots, I_B$ ($|I_k| = n, k = 1, 2, \ldots, B$) are chosen from I_0 randomly according to sample weights $\{w_i^S\}$ and $J_1, J_2, \ldots, J_B$ ($|J_k| \sim m/2, k = 1, 2, \ldots, B$) are chosen from J_0 randomly according to feature weights $\{w_j^F\}$. The weights $\{w_i^S\}$ are updated according to previous misclassification and weights $\{w_j^F\}$ are updated according to the degree of importance on the basis of I_k selected at the current stage. The combine weights $\{\beta_k\}$ are taken as in the same as `Adaboost`.

Weighting in `SF-Boost` algorithm is performed in two directions. One direction is the same as `Adaboost`, that is, weighting is made on the sample set. Another direction is made on the feature set. By default, half features are chosen randomly as the same as in `RSM`, but the selection weights are calculated on the basis of a certain sample subset selected in the resampling stage. Since, in each updating stage, a resampled sample subset is of many of previously-misclassified samples, the features are chosen so as to keep separability on the resampled subset.

The feature weight w_j^F is evaluated on the basis of class separability in jth coordinate. We construct a histogram in which the jth coordinate is divided into a fixed number of intervals staring from the minimum value and ending in the maximum value of samples. Then we measure the mutual information as follows. Let I be the sample subset chosen currently. For jth feature, represented as a set $\{j\}$, we measure the mutual information:

$$H(Y_I) - H(Y_I|X_{I,\{j\}}),$$

where $H(Y_I)$ is the entropy of $Y_I = \{y_i|i \in I\}$ and $H(Y_I|X_{I,\{j\}})$ is the conditional entropy of Y_I given $X_{I,\{j\}}$. The latter is the sample-averaged conditional entropy over short intervals.

The concrete algorithm is given in Fig. 1. It is noted that we can simulate both `Adaboost` and `RSM` easily by activate/inactivate some parts of the algorithm.

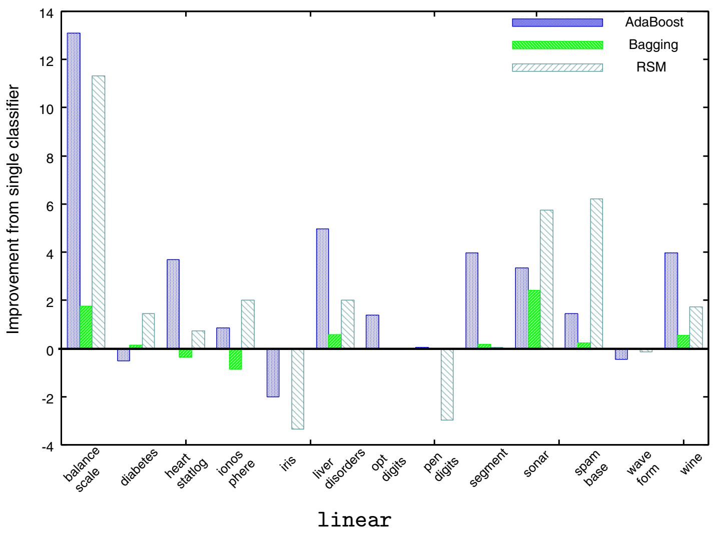
A similar algorithm is already proposed [12]. In the algorithm, a feature selection procedure is employed to find a sub-optimal feature subset for a chosen sample subset. In the goal of feature selection, the procedure is stronger than ours. However, our algorithm has the following advantages: 1) even if the same sample set is chosen repeatedly, the proposed algorithm can choose different features according to its random nature as the same as `RSM`. This is useful to have a necessary number of different classifiers as we hope, and 2) almost all feature selection algorithms cost too much for high-dimensional data. The random selection adopted here is far less in computation time.

```
$X$ : training sample set of size $n$; $X = \{x_{i,j} | i = 1, 2, \ldots, n, j = 1, 2, \ldots, m\}$
$I$ : index set of samples ($I \subseteq I_0 = \{1, 2, \ldots, n\}$)
$J$ : index set of features $J \subseteq J_0 = \{1, 2, \ldots, m\}$)
$X_{I,J}$ : part of $X$ specified by $I$ (samples) and $J$ (features)
$w_i^S$ : weight for $i$th sample
$w_j^F$ : weight for $j$th feature
$A$ : learning algorithm for obtaining a classifier
$\phi$ : component classifier
$\Phi$ : combined classifier
$T$ : maximal number of component classifiers (default: 50)
$error(t)$ : error rate of constructed classifier at step $t$
$error(t) = \frac{1}{n} \sum_{i=1}^{n} I\{y_i \neq \Phi_t(x_i)\}$
$n_b$ : the number of bootstrapped samples (default: $n_b = n$)
$m_b$ : the number of randomly chosen features (default: $m_b = m/2$)
$\sigma$ : convergence precision (default: $\sigma = 0.1$)

Procedure SF-boost($X$: training set, $T, n_b, m_b$) to return a combined classifier $\Phi$

1.  Initialize: $w_i^S \leftarrow 1/n; w_j^F \leftarrow 1/m$ for every $i$th sample and $j$th feature; $t \leftarrow 0$;
2.  While($t < T$){
3.      $I \leftarrow$(randomly chosen $n_b$ samples with replacement according to $\{w_i^S\}$)
4.      (**reweighting of features**)
    $w_j^F \leftarrow H(Y_I) - H(Y_I | X_{I,\{j\}})$ ($j = 1, 2, \ldots, m$)
5.      Normalize $\{w_j^F\}$ to the sum of one.
6.      $J \leftarrow$ (randomly chosen $m_b$ features according to $\{w_j^F\}$)
7.      $\phi_t \leftarrow A(X_{I,J})$   /* construction of a component classifier*/
8.      $\epsilon_t \leftarrow \sum_{i:\phi_k(x_i) \neq y_i} w_i^S$   /* error with sample weights*/
9.      $\beta_t \leftarrow \frac{1}{2} \ln(1 - \epsilon_t)/\epsilon_t$   /* classifier weight */
10.     (**reweighting of samples** )
    If $\phi_t(x_i) \neq y_i$ then $w_i^S \leftarrow w_i^S e^{\beta_t}$; otherwise $w_i^S \leftarrow w_i^S e^{-\beta_t}$ ($i = 1, 2, \ldots, n$)
11.     Normalize $\{w_i^S\}$ to the sum of one.
12.     $\Phi_t = \arg\max \sum_{k=1}^{t} \beta_k \phi_k$
13.     If $|error(t) - error(t-1)| < \sigma$, then exit from the loop.
14.     $t \leftarrow t + 1$
15. }
16. $\Phi(x) = \arg\max_y \sum_t \beta_t \phi_t(x)$
```

Fig. 1. SF-boost algorithm

5 Experiments

We conducted experiments on 13 real-world datasets in UCI repository [13]. For learning algorithms, we used `linear` as a weak algorithm and `SVM` as a strong algorithm. Here *weak* means that the training error is usually not small. We compared four fusion methods: `Adaboost`, `Bagging`, `RSM` and `SF-Boost`. The number of component classifiers is set to $B = 50$. Actually, the performance of most methods was almost converged until 50.



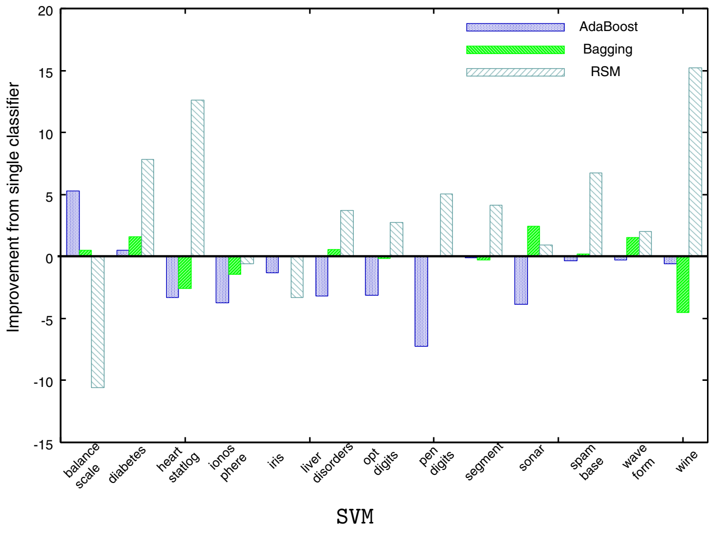


Fig. 2. Comparison of three fusion methods for `linear` and `SVM`. The improved value of the recognition rate from a single classifier is shown. In each dataset, from left, `Adaboost`, `Bagging` and `RSM`.

5.1 Boosting + Weak Classifier vs. Bagging + Strong Classifier

First, we examined the relationship between fusion methods and learning algorithms. The results are shown in Fig. 2. From Fig. 2, it is clear that `linear` was improved by boosting (`Adaboost`) more than by bagging (`Bagging` and `RSM`), and `SVM` was improved by bagging rather than by boosting. These observations are consistent with our prediction. It is also noted that `RSM` outperforms `Bagging` in most cases. Therefore, in the following, we mainly consider the combinations of `Adaboost+linear` and `RSM+SVM`. Both combinations showed a comparable performance as shown in the following Table 1.

5.2 Effectiveness of the Proposed Boosting Algorithm

Next, we examined the effectiveness of our boosting algorithm in which boosting and bagging are coupled. The results are shown in Table 1.

We notice first that our `SF-boost` works better than the others in many cases, regardless of the strengths of learning algorithms. This means that `SF-boost` succeeded to find the most appropriate complexity in these methods. Although it is not shown here, we confirmed that `SF-boost` was almost always better than `Adaboost` (19 of 26 cases). On the other hand, `SF-boost` was better than `RSM` in 16 of 26 cases.

Table 1. Comparison of recognition rate in `AdaBoost` and `SF-Boost`

	Base learning algorithm					
	Linear			SVM		
	Fusion algorithm			Fusion algorithm		
Dataset	Single	AdaBoost	SF-Boost	Single	RSM	SF-Boost
balance-scale	70.91	84.02	**85.93**	**93.60**	83.03	82.87
diabetes	73.43	72.91	**73.68**	63.02	**70.82**	70.71
heart-statlog	72.59	76.29	**76.67**	61.11	73.70	**77.04**
ionosphere	87.16	**88.02**	87.74	**94.30**	93.72	89.44
iris	**98.67**	96.67	96.00	**98.00**	94.67	96.00
liver-disorders	62.60	67.56	**69.28**	62.32	**66.02**	64.87
optdigits	94.40	**95.78**	95.30	96.48	**99.20**	98.51
pendigits	86.90	**86.95**	81.37	77.11	82.17	**97.30**
segment	81.60	85.58	**87.92**	92.46	96.62	**97.32**
sonar	72.12	75.48	**79.33**	79.38	80.33	**81.28**
spambase	68.83	70.29	**90.39**	86.33	**93.04**	92.70
waveform-5000	**86.04**	85.58	85.56	83.82	**85.82**	84.38
wine	74.74	78.72	**92.68**	74.12	89.31	**89.90**
Average	79.23	81.83	**84.76**	81.70	85.26	**86.33**

5.3 On Generalization Ability

It is also interesting to check the generalization ability of those classifiers that can be measured by the difference between the training and testing errors. The smaller the difference is, the higher the generalization ability is. With classifiers of high generalization ability, we can rely on the training error for estimating the testing error. The results are shown in Fig. 3.

In Fig. 3, we notice that 1) weak algorithms (`linear` in our case) have high generalization ability compared with strong algorithms (`SVM` in our case), and 2) among boosting methods, `SF-boost` has higher generalization ability than

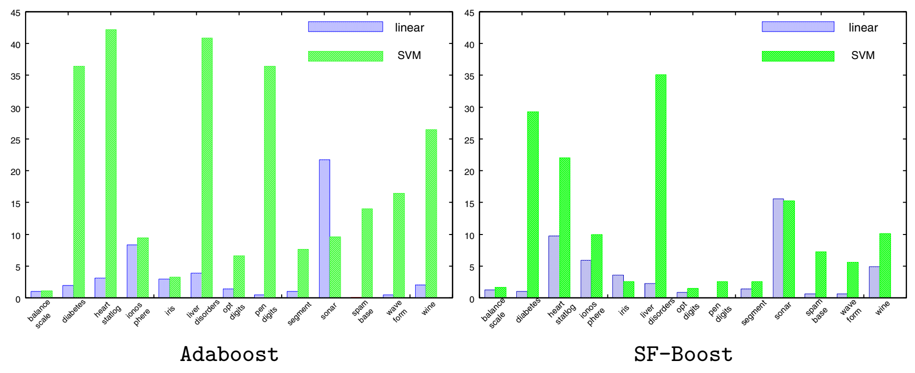


Fig. 3. Difference of testing error and training error by `Adaboost` and `SF-Boost`. In each dataset, `linear` (light) and `SVM` (dark).

`Adaboost`. The second observation implies that a random feature selection in `SF-boost` succeeded to suppress an excessive fitting to the training samples that are often observed in `Adaboost`.

These things tell us that we should use `linear` with some boosting method if we want reliable testing performance, because a high value of recognition rate on the training samples guarantees the same degree of recognition rate for testing samples.

6 Discussion

Obviously, from a limited number of datasets, we cannot derive strong conclusions, but we could have several evidences from experiments.

- It can been seen that boosting is most effective with weak learning algorithms. On the contrary, bagging, especially, bagging using different feature subsets, works with strong algorithms.
- The proposed boosting algorithm in which boosting and feature-based bagging are coupled works well compared with methods on either side.
- For classifier selection, boosting with weak algorithms is most promising because the testing error is close to the training error.

7 Conclusion

We have compared boosting with bagging, especially bagging in different feature subsets, in weak and strong learning algorithms. As a result, it was shown that combinations of (boosting+weak algorithms) and (bagging+strong algorithms) were both better combinations. In addition, we have proposed a boosting method with characteristics of both boosting and bagging and confirmed the advantages on 13 real-world datasets.

There are still many open problems. The first one is to know to what degree the strength of learning algorithms is related to the effectiveness of boosting or bagging. The others are about how to control the balance between boosting and bagging and how to measure the generalization ability of combined classifiers for classifier selection.

References

1. Breiman, L.: Bagging predictors. Machine Learning Journal 2 24, 123–140 (1996)
2. Freund, Y., Shapire, E.: Experiments with a new boosting algorithm. In: Proc. 13th Int. Conf. on Machine Learning, pp. 148–156 (1996)
3. Ho, T.: The random Subspace method for constructing decision forests. IEEE Trans. on Pattern Analysis and Machine Intelligence 20, 832–844 (1998)
4. Friedman, J.: On Bias, Variance, 0/1-Loss, and the Curse-of-Dimensionality. Data Mining and Knowledge Discovery 1, 55–77 (1997)
5. Oh, S.: On the relationship between majority vote accuracy and dependency in multiple classifier systems. Pattern Recognition Letters 24, 359–363 (2003)
6. Ueda, N., Nakano, R.: Analysis of Improvement Effect for Generalization Error of Ensemble Estimators. IEICE technical report. Neurocomputing 95, 107–114 (1996) (in Japanese)
7. Skurichina, M., Duin, R.: Bagging, boosting and the random subspace method for linear classifiers. Pattern Analysis and Applications 5, 121–135 (2002)
8. Shirai, S., Kudo, M., Nakamura, A.: Bagging, Random Subspace Method and Biding. LNCS, vol. 5342, pp. 801–810. Springer, Heidelberg (2008)
9. Skurichina, L., Kuncheva, L.: An Experimental Study on Diversity for Bagging and Boosting with Linear Classifiers. Information Fusion 3, 245–258 (2002)
10. Quinlan, J.: Bagging, boosting, and c4.5. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 725–730 (1996)
11. Kim, H., Pang, S., Je, H., Kim, D., Bang, S.: Pattern Classification Using Support Vector Machine Ensemble. In: Proc. 16th International Conference on Pattern Recognition, 2002, vol. 2, pp. 160–163 (2002)
12. Redpadth, D.B., Levart, K.: Boosting Feature Selection. LNCS, vol. 3086, pp. 305–314. Springer, Heidelberg (2005)
13. Blake, C., Merz, C.: UCI repository of machine learning databases (1998), `http://www.ics.uci.edu/~mlearn/mlrepository.html`

Multi-class Boosting with Class Hierarchies

Goo Jun and Joydeep Ghosh

Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin TX-78712, USA
{gjun,ghosh}@ece.utexas.edu

Abstract. We propose AdaBoost.BHC, a novel multi-class boosting algorithm. AdaBoost.BHC solves a C class problem by using $C-1$ binary classifiers defined by a hierarchy that is learnt on the classes based on their closeness to one another. It then applies AdaBoost to each binary classifier. The proposed algorithm is empirically evaluated with other multi-class AdaBoost algorithms using a variety of datasets. The results show that AdaBoost.BHC is consistently among the top performers, thereby providing a very reliable platform. In particular, it requires significantly less computation than AdaBoost.MH, while exhibiting better or comparable generalization power.

1 Introduction

Adaptive reweighting and combining methods such as boosting have become very popular because of their remarkable ability to reduce both model bias and variance as compared to a base learner. In particular, AdaBoost [1] has been successfully applied to vast range of machine learning applications. AdaBoost is an ensemble learning method for binary classification problems based on a set of weak learners trained under different distributions. There is one baseline requirement for the boosting procedure to work: the weak learner should be at least 50% accurate.

AdaBoost.M1 [2] is a direct extension of the binary AdaBoost algorithm with a multi-class weak learner. The problem of AdaBoost.M1 is that the multi-class weak learner needs to be much stronger, since a random guess would yield only $1/C$ accuracy for a C class problem. This observation has prompted a plethora of approaches that convert a multi-class problem into multiple binary classification problems thus omitting the necessity of directly employing a multi-class classifier. These algorithms either change a single multi-class problem into multiple binary class problems [3], into one big binary class problem with increased number of examples[4], or into a sequence of binary class problems with output codes [5][6].

We take a different approach by employing a binary hierarchical classifier (BHC), which converts a multi-class problem into a set of binary problems based on affinity between classes. In BHC, similar classes are clustered together into meta-classes so that the resulting binary problems are simpler to solve than in other algorithms mentioned above. In this paper, AdaBoost.BHC, a novel method that combines binary AdaBoost with BHC, is proposed and applied to multi-class classification problems, and the results are compared to the results from existing multi-class AdaBoost algorithms. The results show that the performance of AdaBoost.BHC is always among the best, and it runs much faster than AdaBoost.MI and AdaBoost.MH.

2 Dealing with Multi-class Problems

In binary classification, a classifier maps the input space onto a binary output space, $\{+1, -1\}$. In many cases, however, we have to deal with $C > 2$ classes, so the output space is defined as $\{1, 2, ..., C\}$. There are classification algorithms that can directly handle multiple classes, such as decision trees or multi-layer perceptrons. Producing a non-binary output, however, is not possible or less natural for other approaches such as SVMs. In such a case, we can model a multi-class problem using a fusion of binary classifiers.

One way to employ binary classifiers for a multi-class problem is using binary codes to decompose the problem's output space. One-versus-all method and "all-pairs" method are examples of solving multi-class problem by decomposing output spaces. The error-correcting output code (ECOC) [7] is another example of a binary code approach combined with robust error-correcting coding. All of these algorithms can be considered as two-stage approaches in the sense that multiple binary classifiers are trained independently and combined to produce the final class label at the second level. These methods have another common aspect that the binary classification problems are specified without considering similarity between classes. Therefore, the resulting binary problems can become more problematic, e.g. highly unbalanced in one-vs-all approaches, and leading to complicated, multimodal decision boundaries in ECOCs.

An alternative approach is the binary hierarchical classifier (BHC) [8] that was developed for hyperspectral remote-sensing applications where classes (land cover types) have certain natural groupings, i.e. some classes are more similar to one another than to others. BHC recursively decomposes a multi-class problem into $C - 1$ binary (meta-)class problems, arranged as a binary hierarchical tree. In a BHC tree, similar classes are grouped together to form meta-classes at each inner level of the tree. Since the resulting structure of the BHC tree is determined by affinity between classes, the hierarchy often provides useful insights on the problem domain. More importantly, the resulting binary classification problems tend to be simpler, thereby facilitating both feature extraction/selection and classifier design, and making it easier to satisfy the baseline requirement for boosting weak learners.

At the root node of a BHC tree, the given set of classes is first partitioned into two disjoint sets or meta-classes. The meta-classes are recursively partitioned until the leaves of the decomposition tree are reached where each leaf corresponds to only one of the original classes. Consequently, the number of leaf nodes in the BHC tree equals to the number of classes. An outline of the partitioning

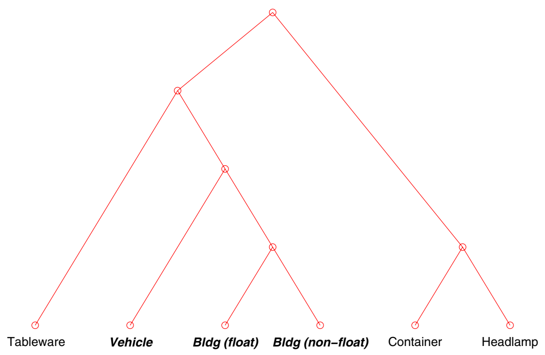


Fig. 1. BHC tree for Glass dataset. Class names in bold italic font are window classes, and others are non-window classes.

Algorithm 1. Outline of PARTITION NODE algorithm

```
1. Initialize associations as $a_1 = 1$, and $a_i = 0.5, \forall i > 1$. $a_i = P(T_l|c_i) = 1 - P(T_r|c_i)$.
2. Find an optimal projection by Fisher's discriminant analysis with soft assignments.
3. Calculate the mean log-likelihood of $T_r$ and $T_l$ in the Fisher-projected space.
4. Update $a_i$'s, using the mean log-likelihoods and a pre-defined step size T.
5. Repeat Steps 2 through 4 until increase in Fisher's discriminant is insignificant.
6. Compute the entropy: $\mathcal{H} = -\frac{1}{M}\sum_i [a_i \log_2 a_i + (1 - a_i) \log_2 (1 - a_i)]$.
7. Stop if $\mathcal{H} < \theta_H$. Otherwise, decrease T and repeat Steps 2 through 6.
```

algorithm is given in Algorithm 1. As a result of the partitioning process, classes that are similar in the input feature space tend to get lumped into the same meta-class in the tree. In the Glass data, for example, all glass types can be divided into two groups: window and non-window. Fig. 1 shows a BHC tree for the Glass data, and we can observe that all classes that belong to the window meta-class are located under the same sub-tree. An empirical study [9] has shown that BHC offers comparable classification accuracies with that of the ECOC with fewer number of classifiers.

Recently, another tree-based approach, margin tree, was proposed [10]. The margin tree algorithm employs the margin between classes as a distance measure for the hierarchical agglomerative clustering of classes. Both BHC and margin tree produce classification trees, but differ in how this tree is built. In the margin tree algorithm, it is assumed that the dimensionality of data is higher than the number of samples, so that the classes are always linearly separable. On the other hand, in BHC we need at least as many samples as number of dimensions, which enables the Fisher's discriminant analysis. In this paper, we chose BHC since all datasets in our experiments have more number of samples than number of features, but our framework is easily applicable to margin tree or other tree-based multi-class decomposition algorithms as well.

3 Multi-class AdaBoost

There are many variations of AdaBoost for multi-class problems. AdaBoost.M1 [2] is a direct extension of the binary AdaBoost algorithm. In AdaBoost.M1, the weak-learner should be able to produce multi-class output. The problem of AdaBoost.M1 is that even weak learning may not be easily obtainable for challenging multi-class problems. Since we are giving more weight to the samples on which our hypotheses fail, the distribution often gets harder to learn for weak learners as we keep boosting, making it much more difficult to produce at least 50% accuracy for multi-class classifiers.

AdaBoost.MH [4] transforms a multi-class problem into a binary classification problem by replicating examples with attached class labels, based on the Hamming loss. AdaBoost.MH can handle both multi-class and multi-label problems. AdaBoost.MH is the most popular multi-class version of AdaBoost in practice [11], and it shows good generalization ability even for relatively hard multi-class problems. One of the problems with AdaBoost.MH is that it converts a multi-class problem into one huge binary problem that requires $N \cdot C$ examples. An alternative approach is AdaBoost.MI [3], a direct application of the one-vs-all method. In AdaBoost.MI, we have C independent weak learners for a C-class problem, and each binary weak learner is dedicated to one class. Each weak learner is trained separately, with independently managed distributions. AdaBoost.MI has similar computational complexity as AdaBoost.MH, but requires a smaller memory footprint because the algorithm can be easily modularized.

Another approach, AdaBoost.OC [5] combines the output code algorithm with AdaBoost. It requires only one binary classifier with N examples, which makes it much faster than other algorithms. AdaBoost.ECC [6] is based on AdaBoost.OC, and it has been shown that AdaBoost.OC is actually a shrinkage version of AdaBoost.ECC [12]. In AdaBoost.ECC, a coloring $\mu_t : Y \rightarrow \{-1, +1\}$ is computed at t-th round of boosting, mapping the output space onto a binary space. A new coloring is obtained for each round, hence we have a sequence of colorings $(\mu_1, \mu_2, ..., \mu_T)$ from T rounds of boosting, which makes each class label correspond to a unique codeword, *e.g.*, (+1, +1, -1...). The codeword from each class labels is multiplied by the outputs of hypotheses and summed up, and the class label which maximizes the value is selected as the final output.

Recently, a different approach that employs a multi-class weak learner was also proposed, where the minimum accuracy requirement for the multi-class weak learner is $1/C$ rather than $1/2$ [11], which is not included in our experiments since we focus on algorithms that work with binary weak learners. We compare performances of MH, MI, and ECC algorithms with that of the proposed AdaBoost.BHC.

4 AdaBoost.BHC

In the AdaBoost.BHC algorithm, a standard binary AdaBoost algorithm is applied to each internal node of the BHC tree, with separately updated distributions. The final hypothesis of each node, H_k, is the weighted sum of hypotheses,

Algorithm 2. AdaBoost.BHC

```
Given $(x_1, y_1), ..., (x_N, y_N)$ and a BHC tree $T$, where $x_i \in X,\ y_i \in Y = \{1, 2, ..., C\}$.

 – $T_1$ is the root node, and $T_k.C \subset Y$ is a set of classes belong to $T_k$.
 – $T_k.L(= l)$ and $T_k.R(= r)$ are indices of left and right child of $T_k$.
 – If $T_k$ is a leaf node, $l = r = 0$ and $|T_k.C| = 1$.

 1. For each internal node $T_k$,
    (a) Construct $(X_k, Y_k) = \{(x_i, y_i)|y_i \in T_k.C\}$,
    (b) Map $(X_k, Y_k) \rightarrow (X_k, Z_k)$, $z_i \in Z_k = \{+1, -1\}$.

          $z_i(x_i) = \begin{cases} +1 & \text{if } \ y_i \in T_l.C \\ -1 & \text{if } \ y_i \in T_r.C \end{cases}$

    (c) Run AdaBoost on with $(X_k, Z_k)$ to obtain $H_k$.
 2. Get $H_{final}(x_i)$, starting from $k = 1$
    (a) If $|T_k.C| = 1$, output $H_{final}(x_i) = y \in T_k.C$ and finish.
    (b) If $H_k(x_i) = +1$, $k = T_k.L$. Otherwise $k = T_k.R$. Return to step 2-(a).
```

and the final multi-class output, $H_{final}(x_i)$ is determined from binary labels generated at each node, $H_k(x_i)$ of the BHC tree. A detailed description of the AdaBoost.BHC algorithm is given in Algorithm 2.

One of the main differences of AdaBoost.BHC from other multi-class AdaBoost algorithms is that the binary decomposition of AdaBoost.BHC is determined from the distribution of the input data. As a result, AdaBoost.BHC produces more separable binary classification problems than other approaches, hence supporting good generalization behavior. Another notable advantage of AdaBoost.BHC is that it requires less number of computations per round than AdaBoost.MH or AdaBoost.MI. The weak learner at the root node is trained with N examples, and the left and the right child nodes of the root are trained with $N/2$ examples on average. Assuming a full balanced binary tree, computational complexity of AdaBoost.BHC algorithm is:

$$\sum_{k=0}^{log_2 C-1} \frac{1}{2^k} f\left(\frac{N}{2^k}\right) ,$$

where $f(\cdot)$ is the complexity of the weak learning algorithm. Table 1 shows computational requirements of MH, MI, ECC, and BHC algorithms for $O(N)$ and $O(N^2)$ weak learning algorithms.

5 Experiments and Results

Empirical comparisons of existing multi-class AdaBoost algorithms and AdaBoost.BHC are made using datasets from the UCI machine learning repository [13] as in Table 2. Seven different multi-class datasets from the repository are

Table 1. Time complexity comparison (per round) between different multi-class AdaBoost algorithms for N examples from C classes

	AdaBoost.MH	AdaBoost.MI	AdaBoost.ECC	AdaBoost.BHC
$O(N)$ weak learner	$C \cdot N$	$C \cdot N$	N	$\log_2 C \cdot N$
$O(N^2)$ weak learner	$C^2 \cdot N^2$	$C \cdot N^2$	N^2	$2(1 - \frac{1}{C})N^2$

Table 2. Datasets used in experiments from UCI repository

Name	# Train	# Test	# Attributes	# Classes
Iris	150	4-CV	4	3
Glass	214	4-CV	10	6
Yeast	1484	4-CV	8	10
Page blocks	5473	4-CV	10	5
Landsat	4435	2000	36	6
Optical digits	3823	1797	64	10
Pen-based digits	7494	3498	16	10

employed. 4-fold cross validation (CV) is done 10 times for Landsat, Optical digits, and Pen-based digits datasets. For the other four datasets with pre-specified test sets, each test is repeated 40 times and the results are averaged. 100 rounds of boosting are done for each algorithm. MATLAB's built-in classification and regression tree (CART) is used as the base learner. Four different algorithms are tested: AdaBoost.MH, AdaBoost.MI, AdaBoost.ECC, and AdaBoost.BHC.

Figs. 2 to 8 show training error and test error of each algorithm for seven datasets. All algorithms except AdaBoost.MI generally show good performance on all data. AdaBoost.BHC and AdaBoost.MH display the best generalization behaviors for most datasets. Note that AdaBoost.BHC achieves low error rates much faster than AdaBoost.MH in most experiments. AdaBoost.MI shows problems with Yeast and Page blocks datasets as shown in Figs. 4 and 5, where both training error and test error increase after some rounds. One probable

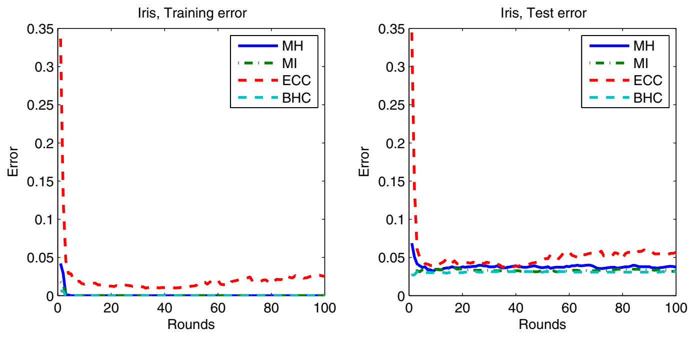


Fig. 2. Training and test errors for Iris data

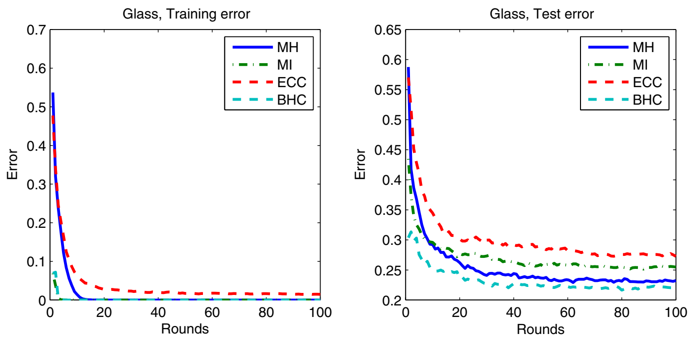


Fig. 3. Training and test errors for Glass data

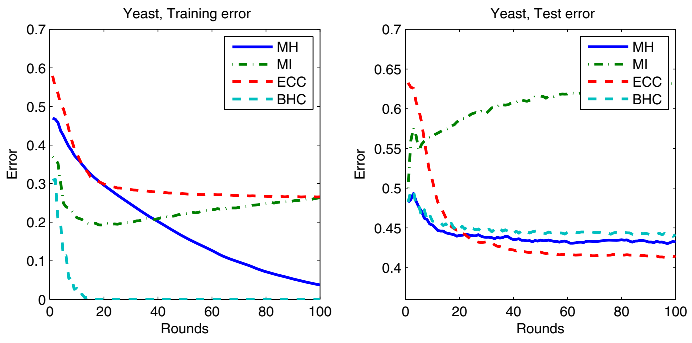


Fig. 4. Training and test errors for Yeast data

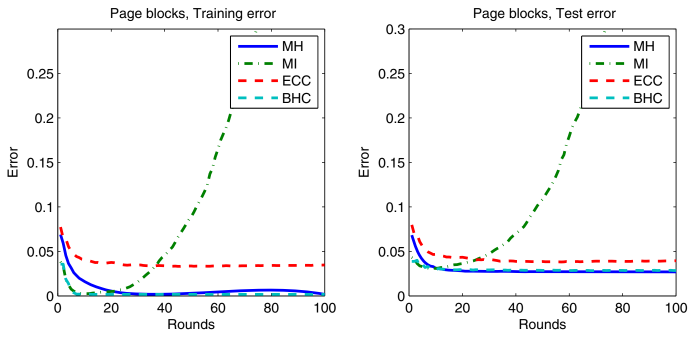


Fig. 5. Training and test errors for Page blocks data

reason is the fact that the class distributions of Yeast and Page blocks datasets are highly unbalanced. The smallest class of the Yeast data has only 5 examples from a total of 1484, and the largest class of the Page blocks dataset has 4913 examples from a total of 5473. Because AdaBoost.MI makes C different

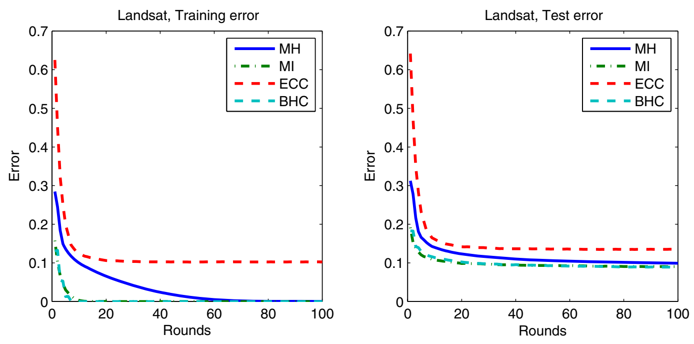


Fig. 6. Training and test errors for Landsat data

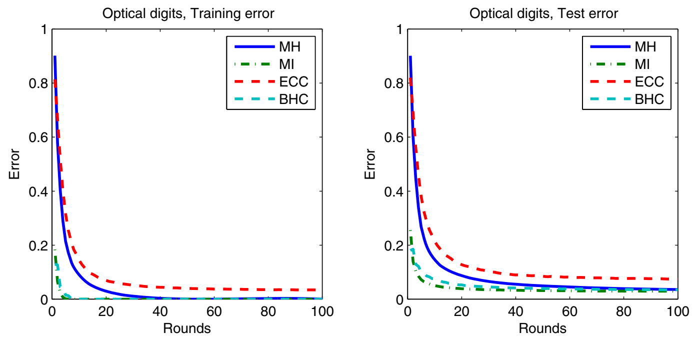


Fig. 7. Training and test errors for Optical digits data

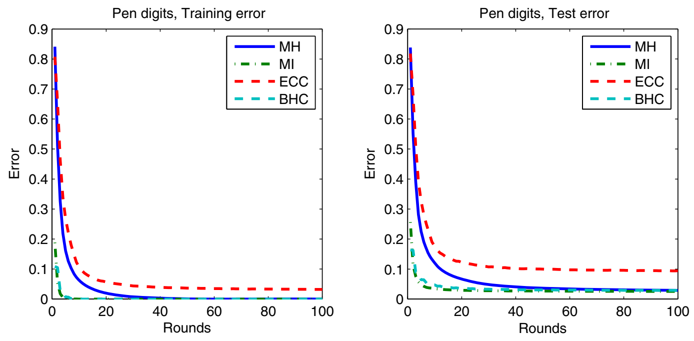


Fig. 8. Training and test errors for Pen digits data

binary classification problems, the highly skewed prior distribution makes the problem much more difficult for weak learners. AdaBoost.MH is less affected by unbalanced distributions, because it converts the hypothesis space from $h : X \rightarrow Y$ to $h : (X, Y) \rightarrow (correct, incorrect)$ domain, hence it always yields the same

Table 3. Average training time(seconds) for 100 rounds

Algorithm	Yeast	Page blocks	Landsat	Optical	Pen digits
AdaBoost.MH	32.0	156.9	311.8	698.9	866.8
AdaBoost.MI	43.5	128.1	331.0	731.4	688.4
AdaBoost.ECC	5.5	15.8	42.5	60.0	33.9
AdaBoost.BHC	24.8	49.3	167.1	198.7	138.1

fraction of positive and negative examples. AdaBoost.ECC and Adaboost.BHC also perform better than AdaBoost.MI under skewed distributions because they aggregate several classes together based on the coding scheme or the affinity between classes. There are more systematic approaches to evaluate performances of classifiers with respect to the complexity of the problem [14], which is left as a future work.

AdaBoost.ECC generally shows relatively higher generalization error except for the Yeast dataset. One possible reason for this is the sub-optimal output space partitioning of the AdaBoost.ECC algorithm. AdaBoost.ECC require mappings from Y to $\{+1, -1\}$ for each round, and a recent study [15] has shown that appropriate partitioning of the output space is important for the algorithm's performance. Here we employed a random partitioning, as suggested in [5]. It is also shown that random partitioning is generally better than the optimized max-cut algorithm, but it still has room for improvement [15]. Our empirical results suggests that Adaboost.BHC provides better output decomposition than AdaBoost.ECC, because it decomposes the output space based on the class-conditional distributions, producing simpler decision boundaries for binary weak learners.

Table 3 shows average training time of different algorithms for large ($N \geq 1000$) datasets. AdaBoost.ECC is clearly the fastest algorithm. AdaBoost.BHC is slower than AdaBoost.ECC, but is significantly faster than AdaBoost.MI and AdaBoost.MH.

6 Conclusions

In this paper, the AdaBoost.BHC algorithm incorporating multiple binary classifiers, a class hierarchy, and boosting was proposed and tested. AdaBoost.BHC is always among the best performers if not the very best, thus providing a more reliable solution. Moreover it is faster than all algorithms other than AdaBoost.ECC. AdaBoost.ECC however typically does not generalize as well as AdaBoost.BHC.

Acknowledgements. This research was supported by NSF Grant IIS-0705815.

References

1. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory, pp. 23–37 (1995)

2. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
3. Abney, S., Schapire, R., Singer, Y.: Boosting applied to tagging and pp attachment (1999)
4. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37(3), 297–336 (1999)
5. Schapire, R.E.: Using output codes to boost multiclass learning problems. In: ICML 1997: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 313–321 (1997)
6. Guruswami, V., Sahai, A.: Multiclass learning, boosting, and error-correcting codes. In: COLT 1999: Proceedings of the twelfth annual conference on Computational learning theory, pp. 145–155. ACM, New York (1999)
7. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artifical Intelligence Research 2, 263 (1995)
8. Kumar, S., Ghosh, J., Crawford, M.M.: Hierarchical fusion of multiple classifiers for hyperspectral data analysis. Pattern Analysis & Applications 5(2), 210–220 (2002)
9. Rajan, S., Ghosh, J.: An empirical comparison of hierarchical vs. two-level approaches to multiclass problems. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 283–292. Springer, Heidelberg (2004)
10. Tibshirani, R., Hastie, T.: Margin trees for high-dimensional classification. J. Mach. Learn. Res. 8, 637–652 (2007)
11. Zhu, J., Rosset, S., Zou, H., Hastie, T.: Multi-class adaboost. Tech. rep., Department of Statistics, University of Michigan, Ann Arbor, MI 48109 (2006)
12. Sun, Y., Todorovic, S., Li, J.: Unifying multi-class adaboost algorithms with binary base learners under the margin framework. Pattern Recogn. Lett. 28(5), 631–643 (2007)
13. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html
14. Ho, T.K., Basu, M.: Complexity measures of supervised classification problems. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 289–300 (2002)
15. Li, L.: Multiclass boosting with repartitioning. In: ICML 2006: Proceedings of the 23rd international conference on Machine learning, pp. 569–576. ACM, New York (2006)

Hybrid Hierarchical Classifiers for Hyperspectral Data Analysis

Goo Jun and Joydeep Ghosh

Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin TX-78712, USA
{gjun,ghosh}@ece.utexas.edu

Abstract. We propose a hybrid hierarchical classifier that solves multi-class problems in high dimensional space using a set of binary classifiers arranged as a tree in the space of classes. It incorporates good aspects of both the binary hierarchical classifier (BHC) and the margin tree algorithm, and is effective over a large range of (sample size, input dimensionality) values. Two aspects of the proposed classifier are empirically evaluated on two hyperspectral datasets: the structure of the class hierarchy and the classification accuracies. The proposed hybrid algorithm is shown to be superior on both aspects when compared to other binary classification trees, including both the BHC and the margin tree algorithm.

1 Introduction

Multi-class problems involving $C > 2$ classes are often tackled using a collection of binary classifiers. One way to employ binary classifiers for multi-class problems is by using binary codes to decompose the problem's output space. Error-correcting output code (ECOC) is a good example of the binary code approach [1]. Other popular approaches include the "all-vs-all" method, where a total of $\binom{C}{2}$ classifiers are needed, and "one-versus-rest" method which requires only C classifiers. In "all-vs-all" each binary classifier gets only a fraction of the data for training, while in one-vs-rest, the data seen is skewed towards the "rest" meta-class.

Another notable way to address a multi-class problem is by constructing a hierarchical tree of binary classifiers. Binary hierarchical classifier (BHC) [2] and margin tree [3] both decompose a multi-class problem into a hierarchically constructed set of binary classification problems. In both algorithms, a C class problem is decomposed into $C - 1$ binary classification problems, and each leaf node corresponds to one of the output classes. The tree structure of BHC or margin tree can also be thought of representing a binary codebook. Decomposing a multiple class problem into several binary classification problems with hierarchy has some advantages, since similar classes are grouped together to form a meta-class, hence providing useful insights into the problem itself. The data seen by each classifier is not so skewed as in one-vs-rest, while the amount of data available to train the classifiers progressively decreases from the root (where all data can be used), to the leaves, where only data for the two classes being considered is inspected.

BHC and margin tree are very similar to each other in the way that they model a multi-class problem using a tree of binary classifiers. Both algorithms were developed to be used for high-dimensional multi-class problems. In this paper we show that the strengths of BHC and margin tree are actually complementary to each other. This suggest the possibility of a hybrid approach that is effective for different types of applications spanning a large range of dimensionality versus sample size. We then propose a hybrid hierarchical classifier that exploits both the BHC and margin tree algorithm to construct a binary classification tree. The proposed algorithm is tested with hyperspectral data, and the resulting tree structure and classification accuracies are compared to those of BHC and margin tree algorithms.

1.1 Binary Hierarchical Classifier (BHC)

The binary hierarchical classifier (BHC) [2] was developed for the classification of hyperspectral data. It decomposes a C class problem into $C-1$ binary classification problems using a binary tree. An empirical evaluation has shown that BHC performs comparably with ECOC using fewer binary classifiers [4]. At the root node of a BHC tree, all classes are first partitioned into two disjoint meta-classes, and obtained meta-classes are recursively partitioned until each single class is assigned to its own meta-class. Consequently, the number of leaf nodes in the BHC tree equals to the number of classes. The partitioning process encourages similar classes to remain in the same partition. At each internal node, for a given set of classes Ω, classes belongs to Ω are to be partitioned into two meta-classes: Ω_0 and Ω_1. The association of each class ω_i is represented by the posterior probability for two meta-classes: $P(\Omega_0|\omega_i)+P(\Omega_1|\omega_i)=1$. The outline of the PARTITION NODE(Ω) algorithm [2] is described in Algorithm 1.

Algorithm 1. PARTITION NODE(Ω)

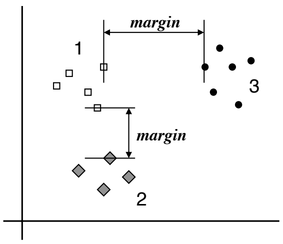
```
1. Initially set $P(\Omega_0|\omega_1)=1$, and $P(\Omega_0|\omega_i)=P(\Omega_1|\omega_i)=0.5$ for $i \neq 1$.
2. Compute the means and covariances of the meta-classes: $\mu_j$ and $\mathbf{\Sigma}_j,\ j \in \{0,1\}$.
3. Compute the fisher projection vector $\mathbf{w}$ with the within-class scatter matrix $\mathbf{S_W}$:
       $\mathbf{w} = \mathbf{S_W}^{-1}(\mu_0 - \mu_1)$ .
4. Compute the mean log-liklihood of meta-classes and update the meta-class posteriors $P(\Omega_j|\omega_i)$ with the log-likelihood and temperature parameter $T$.
5. Repeat steps 2-5 until the incremental increase of the Fisher's discriminant is insignificant.
6. Stop if the entropy of meta-class posteriors is less than the threshold. If not, repeat steps 2-6 after cooling down the temperature $T$.
```

In Algorithm [1], we need the inverse of a $d \times d$ matrix $\mathbf{S_W}$, whose rank cannot exceed the number of samples, n, since it actually is a covariance matrix of sample points. Obviously, $\mathbf{S_W}$ is not invertible when $n < d$, which is called the small sample size problem of Fisher's linear discriminant analysis. One possible and obvious solution is reducing the dimensionality of data by dimensionality reduction techniques. For example, the best-bases feature extraction algorithm [5] aggregates highly-correlated adjacent bands until we have desired number of dimensions suitable for the Fisher's discriminant analysis. Though the best bases algorithm works well with the BHC framework, it is specifically developed for hyperspectral data and not generally applicable to other types of high-dimensional data. Many dimensionality reduction techniques are actually domain-specific, and general methods without any domain knowledge often lead to significant loss of information. Another possible solutions include approximation techniques such as Regularized Discriminant Analysis (RDA) [6] or pseudo-inverse; however these approaches generate inaccurate projections when the scatter matrix is highly singular. Moreover, taking a pseudo-inverse of very high dimensional data is computationally expensive.

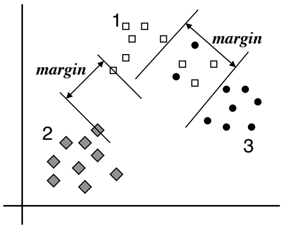
1.2 Margin Tree

In margin tree [3], margins between pairs of classes (or meta-classes) are used as distance measures for clustering of (meta-)classes. There are three different ways to construct a margin tree: greedy, complete-linkage and single-linkage. It is shown that all three methods produce comparable classification results, but the complete-linkage method generates more balanced trees [3]. In this paper, we also employed the complete-linkage margin tree, which is constructed by complete-linkage hierarchical agglomerative clustering (HAC) using margins between classes as distance measures. As a results, a total of $C - 1$ internal nodes will be created with C leaf nodes, same as in BHC.

In the margin tree algorithm, it is assumed that the dimensionality is always greater than the number of samples ($d > n$) [3], so that the samples are always



(a) Linearly separable case



(b) Non-separable case

Fig. 1. In figure (a), class 1 is considered to be closer to the class 2 than to the class 3 because the margin between class 1 and 2 is smaller than the margin between 1 and 3. In figure (b), the margin between class 1 and class 3 is extended due to the misclassified data, thus bigger margin does not necessarily mean more inter-class distance.

linearly separable by a maximum-margin hyperplane. When the samples are not linearly separable, the margin measure is affected by the misclassification cost and the resulting tree structure depends largely on the cost parameter as shown in Fig. 1. Using non-linear kernels such as radial basis function (RBF) is a popular option for SVMs to make the patterns separable in a higher dimensional space; however kernels make the interpretation of margins more difficult, and make the tree structure more sensitive to the kernel parameters.

2 Hybrid Hierarchical Classifier

As described earlier, the small sample size problem occurs in the partitioning process of BHC when we have less number of samples than the dimensionality of data. On the other hand, the weakness of the margin tree is exactly opposite, and the margins between classes are not as meaningful when there is more number of samples than the number of dimensions. Another problem is that the margin is defined only by the samples around the decision boundary, or support vectors. The overall distribution of data is not considered in the tree construction process. In case of the KSC hyperspectral data [2] we used in this paper, the number of samples per class and the number of dimensions are comparable. 13 classes in the KSC dataset are also grouped based on traditional characterisation of vegetation into seven upland and five wetland classes as shown in Table 1. It is often observed in our experiments that BHC fairly distinguishes wetland classes from the upland classes when we have fair amount of training data, while margin tree usually fails to produce meaningful groups.

It is interesting that the requirements for the sample size and the dimensionality from BHC and margin tree are mutually exclusive conditions. We can avoid the small sample size problem by employing the margin tree algorithm whenever we have less number of samples than the number of dimensions, and

Table 1. Landcover classes in the KSC hyperspectral data

Type	Num	Class Name
Upland	1	Scrub
	2	Willow swamp
	3	Cabbage palm hammock
	4	Cabbage Oak hammock
	5	Slash pine
	6	Broadleaf/Oak hammock
	7	Hardwood Swamp
Wetland	8	Graminoid marsh
	9	Spartina marsh
	10	Cattail marsh
	11	Salt marsh
	12	Mud flats
Water	13	Water

Algorithm 2. BUILD TOP-DOWN HYBRID TREE(Ω)

```
1. If $|\Omega| < 3$, return.
2. $S(\Omega) = \sum_{\omega_i \in \Omega} |\omega_i|$
   – If $b \cdot S(\Omega) \leq d + 1$:
       • Build a margin tree with $\Omega$.
   – If $b \cdot S(\Omega) > d + 1$:
       • PARTITION NODE($\Omega$) to get $\Omega_0$ and $\Omega_1$.
       • BUILD TOP-DOWN HYBRID TREE($\Omega_0$).
       • BUILD TOP-DOWN HYBRID TREE($\Omega_1$).
```

we can avoid the linearly inseparable case by applying BHC algorithm whenever the samples are not guaranteed to be linearly separable. In the following sections, two different hybrid algorithms are suggested: top-down and bottom-up.

Building a Top-down Hybrid Tree. In the top-down hybrid method, we initially apply the BHC algorithm starting from the root node, and the margin tree algorithm is called when the partitioned node does not have enough number of training samples. Let $\Omega = \{\omega_1, \omega_2, ..., \omega_C\}$ be the set of all classes, and $S(\Omega)$ be the number of samples in Ω. The top-down hybrid algorithm is described in Algorithm 2. The b value in step 2 is a constant that determines when the transition between BHC and margin tree happens. Larger b means the hybrid tree is closer to a BHC tree, and smaller b means the hybrid tree becomes closer to a margin tree. b should be less than or equal to 1.0 to prevent the small sample size problem. Lower bound of b can be deduced from the Cover's theorem on linear separability [7], according to which random dichotomies are almost surely linearly separable when $b \geq 1.0$. In practice, however, smaller b might be used without any problem since our patterns are not randomly distributed in most cases. Therefore we set $b = 0.5$ in our experiments, and also tested other values ranging from 0.1 to 1.0. Note that when $b \cdot S(\Omega) \leq d + 1$ at the root node, the whole classification tree would be same as the margin tree. On the other extreme, if there are enough number of samples for all classes, then the whole tree would be exactly same as the BHC.

Building a Bottom-up Hybrid Tree. A second way to build a hybrid classification tree is building the tree bottom-up. The overall structure of top-down hybrid tree is close to that of the BHC tree, since initial partitioning of classes at the root node follows the BHC algorithm, unless there are so few number of samples that the tree becomes identical to a margin tree. Unlike the top-down tree, the hybrid tree obtained from the bottom-up approach could have different overall structure from the BHC tree, even at the root node. The main idea is aggregating classes into several meta-classes using the margin tree algorithm until the number of samples in each meta-class becomes sufficient for the BHC algorithm. Once the number of samples in the smallest meta-class (or class) is larger than the dimensionality, we apply BHC on the meta-classes, instead of individual classes. The bottom-up tree building starts with a set of meta-classes, and each

Algorithm 3. BUILD BOTTOM-UP HYBRID TREE(W)

```
1. If $|W| < 2$, stop.
2. Find $\Omega_a$ and $\Omega_b$ $(a \neq b)$ in $W$ s. t.:
       $(\Omega_a, \Omega_b) = \operatorname*{argmin}_{\Omega_i, \Omega_j \in W} \max_{\omega_k \in \Omega_i, \omega_l \in \Omega_j} margin(\omega_k, \omega_l)$
3. Merge $\Omega_a$ and $\Omega_b$ to make a new meta-class $\Omega_c = \Omega_a \cup \Omega_b$.
4. Update the working set: $W = (W - \{\Omega_a\} - \{\Omega_b\}) \cup \{\Omega_c\}$
5. Find $\Omega_m$ and $\Omega_n$ $(m \neq n)$ such that:
       $(\Omega_m, \Omega_n) = \operatorname*{argmin}_{\Omega_m, \Omega_n \in W} S(\Omega_m) + S(\Omega_n)$
   – If $[S(\Omega_m) + S(\Omega_n)] < 2 \cdot d$, then repeat from step 1.
   – Else, build a BHC tree with $W$.
```

meta-class contains only one class in it initially: $\Omega_1 = \{\omega_1\}$, $\Omega_2 = \{\omega_2\}$, ... , $\Omega_C = \{\omega_C\}$. Let's define the working set W as $W = \{\Omega_1, \Omega_2, ..., \Omega_C\}$, initially. The BUILD BOTTOM-UP HYBRID TREE algorithm is shown in Algorithm 3.

3 Experimental Setup

Land cover classification by hyperspectral image (HSI) analysis has become an important part of remote sensing research in recent years [8]. The proposed method was evaluated on hyperspectral images taken from two geographically different locations: NASA's John F. Kennedy Space Center (KSC) [9] and the Okavango Delta in Botswana [10]. We will call the two datasets the KSC and the Botswana datasets, respectively. The KSC dataset was acquired by NASA Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) and originally consisted of 242 bands. After removing noisy bands, only the remaining 176 bands are used. There are 13 different land cover classes including water and mixed classes. The hyperspectral image used for experiments has 512×614 pixels with 18m spatial resolution. The Botswana dataset was obtained from the Okavango Delta by the NASA EO-1 satellite with the Hyperion sensor on May 31, 2001. The acquired data originally consisted of 242 bands, but only 145 bands are used after preprocessing. The area used for experiments has 1476×256 pixels with 30m spatial resolution, with 14 different land cover classes. The KSC dataset has 5121 samples with class labels, and the Botswana dataset has 3248 samples.

For each dataset, samples from each class are randomly divided into 75% training set and 25% test set, and then the training set is sub-sampled to take 10, 20, 30, 50, 75% of the original training set. The purpose of the additional subsampling is to observe the learning curves of different classifiers, and the 100% case is not included since the hybrid algorithms produce results identical to BHC. The random splitting procedure is repeated for 10 times to obtain means and standard deviations of classification accuracies. Each classification tree is trained and tested with two different types of SVM kernels: radial basis function (RBF) and linear. Linear kernel results are included since the original margin tree algorithm is based on the linear kernel [3]. Parameters for RBF kernel is found by 3-fold cross validation for each experiment.

4 Results

Two aspects of the hierarchical classifiers are evaluated: the structure of the class hierarchy, and the classification accuracies. Table 2 and Fig. 2 show the structural evaluations, and Tables 4 and 5 show classification accuracies. Best results are emphasized in bold. Note that we did not compare with "one-vs-all", "all-pairs" or certain direct multi-class methods since a previous, extensive study already showed BHC to be superior than or comparable to these methods for hyperspectral data [4].

Table 2 shows the average distances between different groups of classes generated from the KSC training data, representing how well the hierarchy of the tree complies with the domain knowledge. In Table 1, classes in the KSC data are categorized as upland or wetland classes. Figures in Table 2 are obtained by calculating the minimum tree-traversal distance between two leaf nodes. For example, the distance between two sibling nodes is 2. For each algorithm, the first row indicates the average distance between all upland classes, the second row is the distance between all wetland classes, and the third row indicates the average inter-class distance between each upland and wetland class pairs. A tree is considered to have better structure when the values in the first and the second rows are much smaller than the value in the third row. As can be seen in the table, the bottom-up hybrid algorithm shows more significant separations between two land types with only a small number of training samples. To better visualize this advantage, classification trees obtained from the 20% training samples are shown in Fig. 2. Fig. 2(a) shows the tree structure obtained from the BHC, and trees in Fig. 2(b), 2(c), and 2(d) are from margin tree, top-down hybrid, and bottom-up hybrid algorithms, respectively. Upland class names are printed in normal fonts, and wetland class names are printed in bold italic for distinction. Fig. 2(d) clearly demonstrates significant separation between two groups generated from the bottom-up hybrid hierarchical tree.

Table 4 shows the average classification accuracies and standard deviations from the KSC data, and Table 5 shows the results for the Botswana data. In most cases there are not much difference in classification accuracies. Although not significant, we can still observe the tendency changes as the increased number of data points: top-down hybrid results are closer to the margin tree results when we have smaller number of samples, and the results become closer to the BHC results as we have more training samples. The bottom-up hybrid algorithm shows at least equal performances to all other algorithms, and shows better results than others when the linear kernel is employed.

We also tested the effect of b value in the top-down hybrid algorithm. Table 3 shows the average classification accuracies for the KSC dataset with different

Table 2. Average distances between: 1) upland classes 2) wetland classes 3) upland and wetland classes

Tree	Distance	Training set size (% of full training set)				
		10%	20%	30%	50%	75%
BHC	Upland	5.70 (0.24)	5.61 (0.25)	5.35 (0.20)	5.12 (0.15)	4.97 (0.04)
	Wetland	6.74 (0.33)	6.40 (0.47)	6.12 (0.22)	5.72 (0.22)	5.44 (0.08)
	Between	6.40 (0.26)	6.30 (0.19)	6.37 (0.11)	6.53 (0.07)	6.63 (0.05)
Margin Tree	Upland	6.30 (0.70)	6.02 (0.86)	6.64 (0.50)	6.42 (0.81)	5.60 (0.86)
	Wetland	5.42 (0.55)	5.34 (0.50)	5.40 (0.47)	5.34 (0.49)	5.62 (0.18)
	Between	6.47 (0.40)	6.53 (0.38)	6.39 (0.43)	6.49 (0.45)	6.68 (0.49)
Hybrid Top-down	Upland	6.31 (0.26)	5.72 (0.18)	5.43 (0.28)	5.12 (0.15)	4.97 (0.04)
	Wetland	5.72 (0.34)	6.06 (0.33)	5.96 (0.25)	5.72 (0.22)	5.44 (0.08)
	Between	6.47 (0.25)	6.37 (0.14)	6.37 (0.11)	6.52 (0.07)	6.63 (0.05)
Hybrid Bottom-up	Upland	**4.53** (0.18)	**4.72** (0.60)	**5.20** (0.86)	**5.10** (0.22)	**4.95** (0.00)
	Wetland	**3.82** (0.63)	**4.24** (0.75)	**4.16** (0.48)	**5.46** (0.34)	**5.28** (0.25)
	Between	**8.06** (0.52)	**8.10** (0.89)	**8.09** (0.68)	**6.83** (0.32)	**7.03** (0.19)

Table 3. Classification accuracies for KSC data with different values of b

b	Training set size (% of full training set)					
	10%	15%	20%	30%	50%	75%
0.1	89.45 (1.23)	91.59 (1.02)	**92.45** (0.85)	**93.55** (0.65)	94.38 (0.65)	95.12 (0.50)
0.3	89.52 (1.22)	**91.64** (0.92)	92.25 (0.95)	93.43 (0.71)	**94.75** (0.67)	**95.29** (0.57)
0.5	**89.79** (0.85)	90.80 (1.41)	91.65 (1.10)	93.40 (0.70)	94.73 (0.67)	95.23 (0.49)
0.7	89.32 (1.41)	90.60 (1.53)	91.63 (1.29)	93.43 (0.67)	94.71 (0.59)	95.24 (0.51)
1.0	89.66 (1.42)	90.00 (0.88)	91.58 (1.37)	93.41 (0.62)	94.71 (0.59)	95.24 (0.51)

Table 4. Classification accuracy (%) for KSC data

(a) RBF kernel

Tree	Training set size (% of full training set)				
	10%	20%	30%	50%	75%
BHC	89.66 (3.04)	91.47 (3.70)	93.38 (3.58)	**94.74** (2.70)	95.23 (2.10)
Margin Tree	89.45 (4.66)	**92.45** (1.56)	**93.55** (3.14)	94.37 (3.29)	95.12 (2.42)
Hybrid Top-down	**89.79** (0.85)	91.65 (1.10)	93.40 (0.70)	94.73 (0.67)	95.23 (0.49)
Hybrid Bottom-up	89.17 (0.70)	92.25 (0.90)	93.43 (0.61)	94.41 (0.71)	**95.36** (0.49)

(b) Linear kernel

Tree	Training set size (% of full training set)				
	10%	20%	30%	50%	75%
BHC	85.90 (0.75)	89.76 (1.12)	91.02 (1.44)	93.66 (1.38)	**94.94** (0.60)
Margin Tree	87.20 (1.62)	90.99 (1.72)	89.97 (1.56)	89.80 (0.82)	91.07 (1.55)
Hybrid Top-down	87.20 (2.43)	90.01 (1.13)	91.15 (1.42)	**93.67** (1.38)	**94.94** (0.60)
Hybrid Bottom-up	**90.90** (1.07)	**92.30** 0.77)	**92.82** (1.24)	93.20 (1.26)	94.21 (0.98)

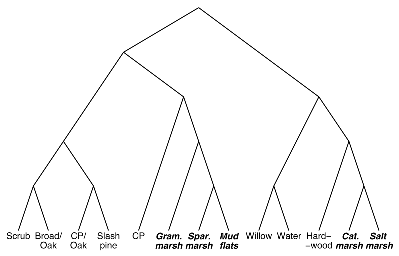
Table 5. Classification accuracy (%) for Botswana data

(a) RBF kernel

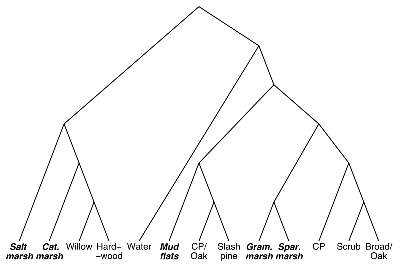
Tree	Training set size (% of full training set)				
	10%	20%	30%	50%	75%
BHC	86.84 (1.84)	**91.14** (1.17)	91.92 (1.41)	**94.97** (0.64)	96.10 (0.90)
Margin Tree	87.84 (2.19)	91.09 (1.44)	**92.94** (0.91)	94.92 (0.82)	96.01 (1.07)
Hybrid Top-down	**87.86** (1.61)	90.63 (1.38)	92.24 (1.08)	**94.97** (0.67)	96.10 (0.90)
Hybrid Bottom-up	87.29 (1.93)	91.11 (0.90)	92.53 (1.01)	94.76 (1.16)	**96.11** (0.92)

(b) Linear kernel

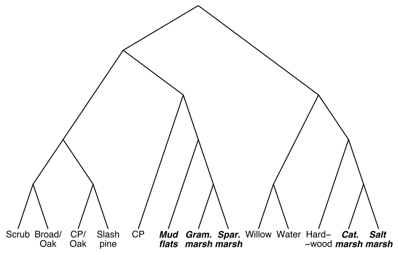
Tree	Training set size (% of full training set)				
	10%	20%	30%	50%	75%
BHC	80.27 (3.33)	87.37 (2.37)	88.91 (1.67)	89.48 (1.63)	90.34 (1.72)
Margin Tree	86.40 (3.76)	87.21 (5.16)	90.31 (2.56)	90.49 (3.78)	**94.18** (3.78)
Hybrid Top-down	82.47 (4.22)	87.66 (2.49)	88.84 (1.82)	89.48 (1.63)	90.34 (1.72)
Hybrid Bottom-up	**90.38** (1.48)	**93.54** (0.68)	**93.36** (1.81)	**91.11** (1.21)	90.09 (1.50)



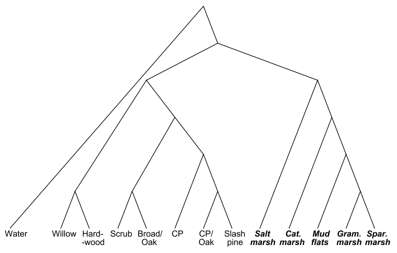
(a) BHC



(b) Margin Tree



(c) Top-down Hybrid



(d) Bottom-up Hybrid

Fig. 2. Typical tree structure from BHC, Margin Tree, Top-down Hybrid, and Bottom-up Hybrid

values of b. When b is very small, the numbers are similar to the results from the margin tree in Table 4, and the results are more similar with the results from the BHC when $b = 1.0$.

5 Conclusion

It is shown that by using the proposed hybrid hierarchical classifiers, a classification tree can be built much more effectively than by using the BHC alone, because the proposed method does not suffer from the small sample size problem. The hybrid algorithm can also generate much more meaningful tree structures than the margin tree can because the suggested method looks for margins between classes only when the samples are linearly separable. The performance of the generated classification tree is evaluated by two measures: the separation between upland and wetland classes, and the classification accuracy. Bottom-up hybrid approach shows best separation results, and both hybrid approach yielded comparable, if not better, classification accuracies.

Acknowledgements. This research was supported by NSF Grant IIS-0705815.

References

1. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artifical Intelligence Research 2, 263 (1995)
2. Kumar, S., Ghosh, J., Crawford, M.M.: Hierarchical fusion of multiple classifiers for hyperspectral data analysis. Pattern Analysis & Applications 5(2), 210–220 (2002)
3. Tibshirani, R., Hastie, T.: Margin trees for high-dimensional classification. J. Mach. Learn. Res. 8, 637–652 (2007)
4. Rajan, S., Ghosh, J.: An empirical comparison of hierarchical vs. two-level approaches to multiclass problems. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 283–292. Springer, Heidelberg (2004)
5. Kumar, S., Ghosh, J., Crawford, M.M.: Best-bases feature extraction algorithms for classification of hyperspectral data. IEEE Trans. on Geosci. and Remote Sens. 39(7), 1368–1379 (2001)
6. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, New York (2001)
7. Cover, T.M.: Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE Transactions on Electronic Computers EC-14(3), 326–334 (1965)
8. Landgrebe, D.: Hyperspectral image data analysis. IEEE Signal Processing Magazine 19, 17–28 (2002)
9. Morgan, J.T.: Adaptive hierarchical classifier with limited training data. Ph.D thesis, Univ. of Texas at Austin (2002)
10. Ham, J., Chen, Y., Crawford, M.M., Ghosh, J.: Investigation of the random forest framework for classification of hyperspectral data. IEEE Trans. Geosci. and Remote Sens. 43(3), 492–501 (2005)

Multiple Classifier Combination for Hyperspectral Remote Sensing Image Classification

Peijun Du, Wei Zhang, and Hao Sun

Department of Remote Sensing and Geographical Information Science,
China University of Mining and Technology,
Xuzhou City Jiangsu Province 221116, P.R. China
dupjrs@cumt.edu.cn, dupjrs@hotmail.com

Abstract. Multiple classifier combination is used to hyperspectral remote sensing image classification in this paper, and some classifier combination algorithms are experimented. Based on a brief introduction to multiple classifier system and general algorithms, a modified evidence combination algorithm is adopted to handle evidence with high inconsistency, and a hierarchical multiple classifier system is designed to integrate the advantages of multiple classifiers. Using the OMIS hyperspectral remote sensing image as the study data, training samples manipulation approaching including boosting and bagging, together with parallel and hierarchical combination schemes are experimented. Mahalanobis distance classifier, MLPNN, RBFNN, SVM and J4.8 decision tree are selected as member classifiers. In the multiple classifier combination scheme based on training samples, both boosting and bagging can enhance the classification accuracy of any individual classifier, and boosting performs a bit better than bagging when the same classifier is used. In classification ensemble using multiple classifier combination approaches, both parallel combination using modified evidence theory and hierarchical classifier system can obtain higher accuracy than any individual member classifiers. So it can be concluded that multiple classifier combination is suitable for hyperspectral remote sensing image classification.

Keywords: Multiple classifier combination, classifier ensemble, evidence theory, boosting, bagging, hyperspectral remote sensing.

1 Introduction

Remotely Sensed image has been widely used to land cover classification and thematic mapping owing to its diverse advantages. Classification accuracy is the most important indicator for remote sensing applications. In order to improve classification accuracy, two strategies are adopted usually: one is to develop novel advanced classifiers, such as artificial neural network [1-4], object-oriented classifier, support vector machine classifier [5-6], decision tree, artificial immune system [7], expert system and so on; the other is to generate a classifier ensemble by combining some individual classifiers for a specific task, which is named as classifier ensemble or multiple classifier system [8]. Decision level fusion based on classification results from different classifiers is a popular way to integrate multiple classifiers, and some fusion algorithms have been developed and employed, for example, majority voting, D-S evidence theory, fuzzy integral, weighted summation, consensus, mixed neural network and hierarchical classifier system [9-11].

Recently, Multiple Classifier System (MCS) has been widely used in such fields as pattern recognition, image processing and target identification. Classifier combination can make use of the complementarity among multiple classifiers and improve the total accuracy. For example, Bo used voting and Bayesian average fusion algorithms to Landsat TM image classification and the total accuracy was higher than any individual classifier [12]; Zhang employed multiple classifier combination to urban vegetation classification from IKONOS image and obtained satisfactory results [13]; Foody used majority voting rule to integrate multiple classifiers for two-class classification [14]; Doan investigated the combination of soft classification methods for remote sensing image and found that classification combination could improve the accuracy [15]; Dehghani applied a hierarchical classifier combination method to hyperspectral remote sensing image [16]□and Benediktsson discussed the use of multiple classifier systems in remote sensing [17]. All those studies have shown the merits and prospects of multiple classifier combination to remote sensing.

In this paper the study is focused on the use of classifier ensemble to hyperspectral remote sensing image classification from two aspects: combination based on sample manipulation, and combination based on multiple classifiers. Using OMIS (Operational Modular Imaging Spectrometer) aerial hyperspectral remote sensing data, the performance of classifier ensemble adopting different combination strategies is experimented and then compared with traditional individual classifier.

2 Multiple Classifier Combination

2.1 Concept

Assume M classes exist on a remote sensing image, and $C_1 \cup C_2 \cup \ldots C_i \ldots \cup C_M$, $i \in \{1, 2, \ldots, M\}$. For hard classification, each pixel is allocated a specific class label J, $J \in \{1, 2, \ldots, M+1\}$. For soft classification, each pixel is depicted by a vector $\{P(1), P(2), \ldots, P(M)\}$, where P(i) is the probability or membership of the pixel belonging to the *i*th class, or the abundance of the *i*th endmember in the pixel.

When multiple classifiers are combined, both the results of hard classification and that of soft classification can be integrated, or the hierarchical tree structure can be used. Some widely used classifier combination methods are majority voting, Bayesian, D-S evidence theory, fuzzy integral and dynamic classification selection.

2.2 Multiple Classifier Combination Based on Sample Manipulation

In recent years, multiple classifier combination methods based on training sample manipulation have got more attention from pattern recognition field, and the most widely used methods are Boosting and Bagging. Unlike statistical voting theory which is based on the assumption of independent data sources and uses all training samples only one time, Boosting and Bagging are exerted by manipulating training samples [18-19].

Bagging is the abbreviation of Bootstrap Aggregating. In this algorithm, *n* samples are selected at random from a set with *k* samples, and instructive iteration is exerted to create some different bags, and every bags is classified by vote to predict its class.

Similar to Bagging, Boosting is also based on the manipulation to training samples. Boosting can process data with weight, so the weights of misclassified samples are increased to concentrate the learning algorithm on specific samples.

In this research, Bagging, Boosting, J4.8 decision tree, MLP neural network, RBF neural network and others are used to form the classifier ensemble for hyperspectral remote sensing image classification. In the experiments, $k_{\max}$=10 and the sample number of every iteration is 75% of the total samples.

2.3 Multiple Classifier Combination Based on Classifier Ensemble

In multiple classifier combination based on classifier ensemble, identical training samples are used to every member classifier to derive its classification result, and then all results are integrated based on a specific algorithm, for example, majority voting, Bayesian average, D-S evidence theory and fuzzy integral. In this experiment, a modified D-S evidence theory and a hierarchical classifier ensemble are used to combine multiple classifiers

2.3.1 D-S Evidence Theory and Its Improvement

Compared with Bayesian theory, D-S evidence theory assigns probability to sets and can handle the uncertainty caused by unknown factors. D-S evidence theory uses discrimination framework, confidence function, likelihood function and probability allocation function to represent and process knowledge [20]. Suppose that $\Theta = \{C_1, C_2, \cdots C_i, \cdots C_M\}$ is discrimination framework and M is the number of classes, therefore basic probability allocation function *m* is a function from 2^{Θ} to [0, 1 and it meets the requirement of:

$$\begin{cases} m(\phi) = 0 \\ \sum_{A \subseteq \Theta} m(A) = 1 \end{cases} \tag{1}$$

If there are two or more different evidences, orthogonal sum can be used to combine those evidences. Assume that $Z_1, Z_2, \ldots, Z_n$ are those probability allocation functions corresponding to evidence $F_1, F_2, \ldots$, and F_n, and their orthogonal sum $Z=Z_1 \oplus Z_2 \ldots \oplus \ldots \oplus Z_n$ is:

$$Z(\phi) = 0 \tag{2}$$

$$Z(A) = K^{-1} \times \sum_{\cap A_i} \prod_{1 \le i \le n} Z_i(A_i) \tag{3}$$

$$K = \sum_{\cap A_i \ne \phi} \prod_{1 \le i \le n} Z_i(A_i) \tag{4}$$

When various evidences are inconsistent or contradictory each other, the combined result of D-S evidence may be unreasonable [21]. A modified evidence combination algorithms was proposed and experimented by Sun et al, and it proved that the modified method was superior to traditional method while processing those evidences with high contradiction and inconsistency [22]. For remote sensing image, different classifier may generate different classified labels, which result in the generation of evidence with high contradiction, so the modified evidence combination is applied to classification integration of hyperspectral remote sensing images. The detailed equations are as follows [22]:

$$k_{ij} = \sum_{\substack{A_i \cap A_j = \phi \\ A_i \in F_i, A_j \in F_j}} Z_i(A_i) Z_j(A_j) \tag{5}$$

$$\tilde{k} = \frac{1}{n(n-1)/2} \sum_{i<j} k_{ij} \tag{6}$$

$$\varepsilon = e^{-\tilde{k}} \tag{7}$$

$$Z(A) = p(A) + K * \varepsilon * q(A), A \neq \Phi, \Theta \tag{8}$$

$$Z(\Theta) = p(\Theta) + K * \varepsilon * q(\Theta) + k(1-\varepsilon) \tag{9}$$

$$p(A) = \sum_{\substack{A_i \in F_i \\ \cap_{i=1}^{n} A_i = A}} Z_1(A_1) Z_2(A_2) \cdots Z_n(A_n) \tag{10}$$

$$q(A) = \frac{1}{n} \sum_{i=1}^{n} Z_i(A) \tag{11}$$

Where, ε is the confidence of evidence, $\tilde{k}$ is the average of contradiction level between two evidences, and K is the total contradiction level of all evidences. This evidence combination method can reduce the limitations caused by high evidence inconsistency.

For multiple classifier combination of remote sensing, the classifier result of each classifier can be viewed as a piece of evidence. Probability allocation function can be represented by the classification accuracy of specific class. For example, if a pixel is classifier to the *i*th class, the basic probability is: $m(C_i)=P_i$, $m(\Theta)=1-P_i$, where P_i is the accuracy of the *i*th class by the specific class. After evidence combination being completed, the class with maximum evidence is selected as the final result.

2.3.2 Hierarchical Classifier Combination

In this hierarchical classifier framework, the probability output from each individual classifier is used as the input of classifiers in the next level. The inaccuracy of class probability and inconsistency of inter-classifiers in the first level can be reduced by processing and classifying the output probability in the second level. Since support vector machines (SVMs) perform well for decision level fusion, SVM classifier is used in the second level, and MLP neural network is used in the first level. Fig.1 is the structure of this proposed hierarchical classifier ensemble.

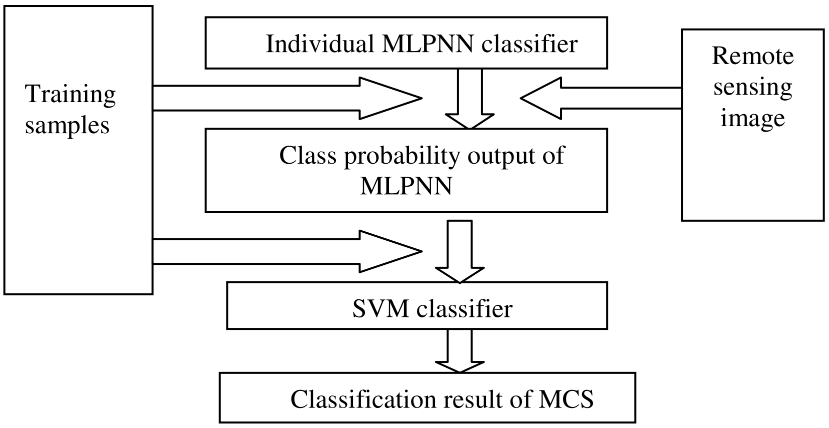

Training samples
Individual MLPNN classifier
Remote sensing image
Class probability output of MLPNN
SVM classifier
Classification result of MCS

Fig. 1. Structure of hierarchical classifier system

3 Hyperspectral Remote Sensing Data

OMIS hyperspectral remote sensing image with 64 bands is classified by above classifier ensemble, and 59 bands out of 64 bands are used for classification. The image is classified into five classes: water, building, vegetation, forest and bare soil. Fig.2 is the false color composite of the image by using Band 27, 25 and 2 as R, G and B components. Training samples and test samples are selected independently from the image, and Table 1 is the information about samples.

Table 1. Information about training samples and test samples

class	Number of training samples	Number of training samples
water	202	327
forest	356	301
vegetation	167	259
building	252	263
bare soil	172	283

4 Experiment and Analysis

The member classifiers in the classifier ensemble include: Mahalanobis distance classifier, MLPNN, RBFNN, SVM classifier, J4.8 decision tree. Those classifiers are based on different principles and criterions. Classifier combination is implemented using IDL language and ENVI software.

In evidence theory method, Mahalanobis distance classifier, MLPNN and SVM classifier are used to form the classifier ensemble, and classification accuracy is evaluated by total accuracy and kappa coefficient. Fig.3 shows the classification results of SVM, J4.8 decision tree, RBFNN and MLPNN respectively. Fig.4 are

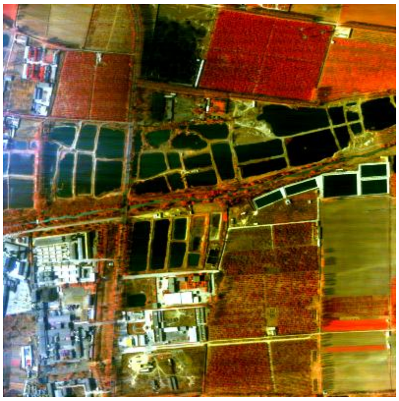
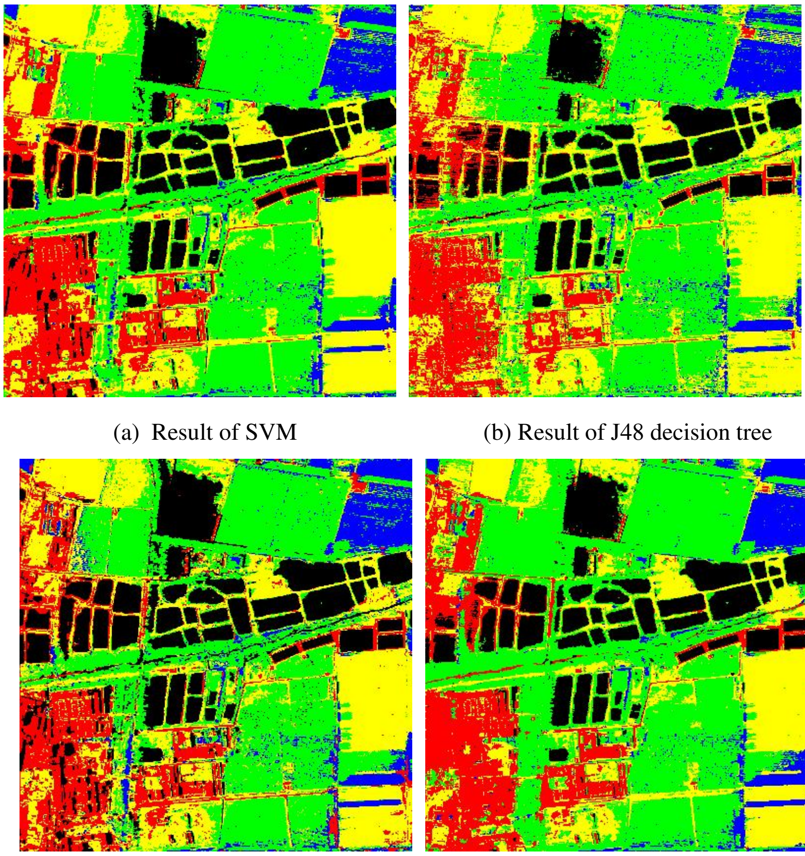


Fig. 2. The false color composite of the hyperspectral remote sensing data



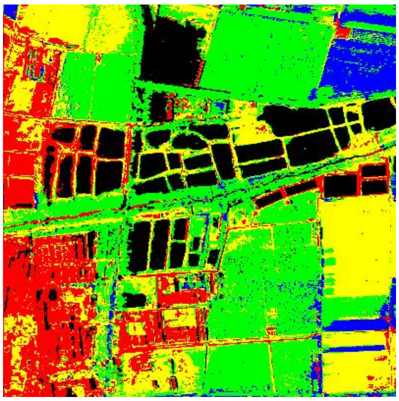
(a) Result of SVM

(b) Result of J48 decision tree

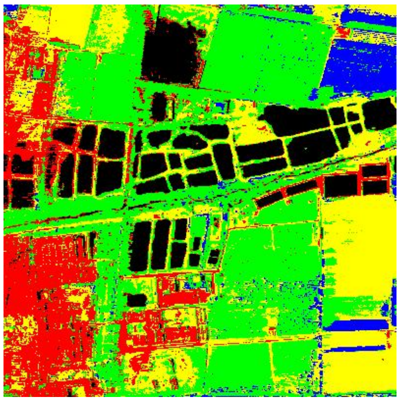
(c) Result of RBFNN

(d) Result of MLPNN

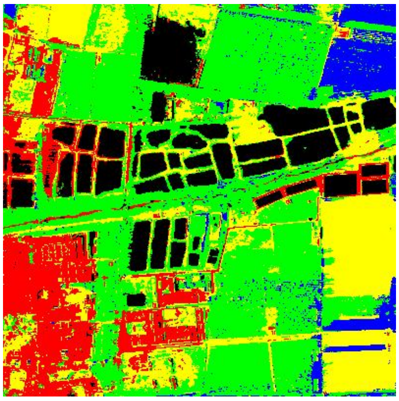
Fig. 3. Classification results of individual classifier



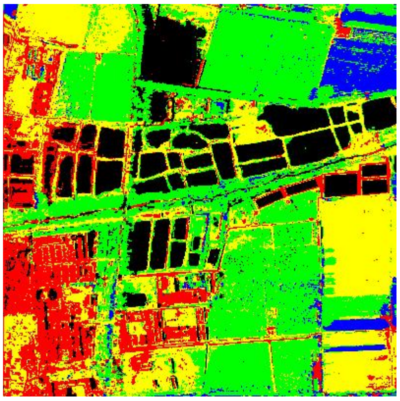
(a) RBFNN based on bagging



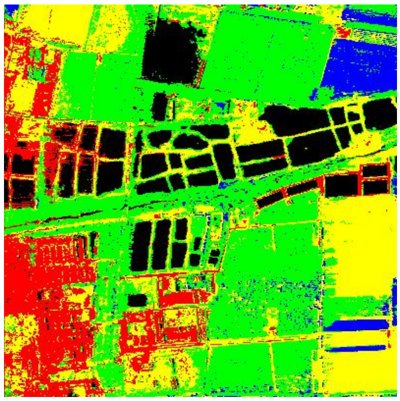
(b) J4.8 decision tree based on bagging



(c) MLPNN based on bagging



(d) RBFNN based on boosting



(e) J48 decision tree based on boosting

⬛	Water
🟥	Building
🟩	Forest
🟦	Vegetation
🟨	Bare soil

Fig. 4. Classification result based on boosting and bagging

the classification results based on boosting and bagging, including RBFNN based on bagging, J4.8 decision tree based on bagging, MLPNN based on bagging, RBFNN based on boosting, J4.8 DT based on boosting. Fig.5 is the result of modified evidence theory, and Fig.6 is the result of hierarchical classifier combination. The classification of single classifiers are listed in Table 2.The accuracy indicators of different classification schemes are listed in Table 3.

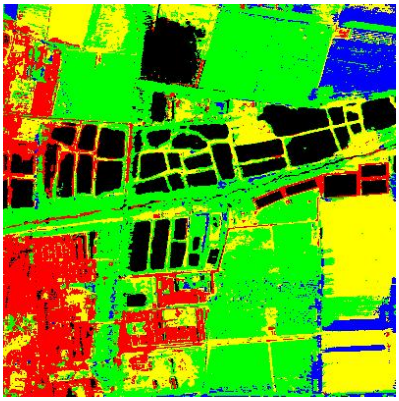


Fig. 5. Classification result of multiple classifier combination based on modified evidence theory

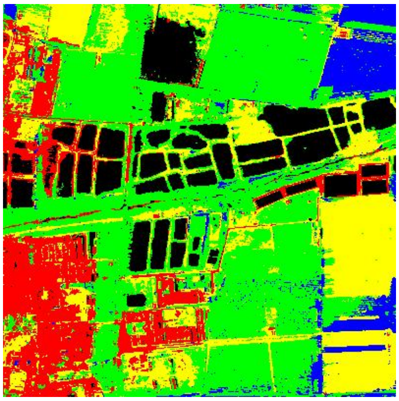


Fig. 6. Classification result of hierarchical classifier combination

In Table 2, SVM classifier has the best classification accuracy of forest and bare soil; MLPNN classifier has the best classification accuracy of vegetation and building; RBF classifier and Mahalanobis distance classifier have the best classification accuracy of water. From Table 3, it can be found that all multiple classifier combination methods can improve the accuracy of hyperspectral remote sensing image classification in contrast with those individual member classifiers.

Table 2. Classification accuracy of single classifier

classifier	Water	Building	Forest	Vegetation	Boil soil
SVM	80.57%	97.34%	**92.69%**	89.96%	**92.97%**
J48 decision tree	88.07%	86.38%	88.80%	95.82%	80.92%
RBFNN	**95.72%**	88.70%	89.58%	90.87%	81.63%
Mahalanobis distance classifier	**94.80%**	96.35%	88.42%	95.82%	80.92%
MLPNN	87.77%	**98.34%**	90.35%	**98.10%**	88.69%

In the multiple classifier combination scheme based on training samples, both Boosting and Bagging can enhance the classification accuracy of any individual classifier. In contrast, boosting performs a bit better than bagging when the same classifier is used.

In classification ensemble based on multiple classifier combination, both modified evidence theory and hierarchical classifier system can obtain higher accuracy than any individual member classifier.

Table 3. Classification accuracy statistics

classifier	Total accuracy	Kappa
SVM	90.7200%	0.8838
J48 decision tree	87.8576%	0.8480
RBFNN	89.4627%	0.8680
Mahalanobis distance classifier	91.4166%	0.8925
MLPNN	92.5331%	0.9065
RBFNN based on bagging	89.8814%	0.8732
MLPNN based on bagging	93.0914%	0.9135
J4.8 DT based on bagging	90.4396%	0.8802
RBFNN based on boosting	90.6490%	0.8829
J4.8 DT based on boosting	90.9979%	0.8872
Modified evidence combination	92.9518%	0.9117
Hierarchical classifier combination	93.5799%	0.9196

5 Conclusion

Using land cover classification from OMIS hyperspectral remote sensing image as an example, the applications of multiple classifier combination to hyperspectral remote sensing is experimented in this paper. Classifier ensemble can improve the classification accuracy of hyperspectral remote sensing image, and it is a potentially important and advanced classification strategy. Both classifier combination based on training samples and multiple classifier combination based on evidence theory and hierarchical system can enhance the classification accuracy obviously. Both hierarchical classifier combination and parallel classification combination can enhance classification accuracy, and their performances are affected by different factors such as selected member classifiers, classifier combination criterion and others.

Acknowledgments. The authors thank the support to this research from National 863 High-Tech Program of China (2007AA12Z162), the Natural Science Foundation of China (40871195), and the Program for New Century Excellent Talents in University (NCET-06-0476).

References

1. Ha, S., Ma, J.W., Li, Q.Q., et al.: Dimension Reduction of Self-organized Neural Network Classification for Multi-band Satellite Data. Geomatics and Information Science of Wuhan University (5), 461–466 (2004)
2. Carpenter, G.A., Gjaja, M.N., Gopal, S.: ART Neural Networks for Remote Sensing: Vegetation Classification from Landsat TM and Terrain Data. IEEE Transactions on Geoscience and Remote Sensing (3), 308–325 (1997)

3. Zhang, Y., Shao, M.Z.: Unmixing Based on Radial Basis Function Neural Network. Journal of Remote Sensing (7), 285–288 (2002)
4. Zhang, W., Du, P.J., Zhang, H.P.: Mixed Pixel Decomposition Based on Neural Network. Bulletin of Surveying and Mapping (7), 23–36 (2007)
5. He, M.L., Sheng, Z.Q., Kong, F.S., et al.: Study on Multi-source Remote Sensing Images Classification with SVM. Journal of Image and Graphics (4), 648–654 (2007)
6. Waske, B., Menz, G., Benediktsson, J.A.: Fusion of Support Vector Machines for Classifying SAR and Multispectral Imagery from Agricultural Areasm. In: IEEE Geoscience and Remote Sensing Symposium, pp. 4842–4845 (2007)
7. Zhang, Y.F., Zhang, L.P., Gong, J.Y., et al.: Remote Sensing Image Classification Based on Artificial Immune System. Journal of Remote Sensing (7), 374–380 (2005)
8. Lu, D., Weng, Q.: A survey of image classification methods and techniques for improving classification performance. International Journal of Remote Sensing 28(3), 823–870 (2007)
9. Steele, B.M.: Combining Multiple Classifiers: An Application Using Spatial and Remotely Sensed Information for Land Cover Type Mapping. In: Remote Sensing of Environment, pp. 545–556 (2000)
10. Ahmed, A., Mohamed, D.: A New Technique for Combining Multiple Classifiers Using the Dempster-Shafer Theory of Evidence. Journal of Artificial Intelligence Research (2002)
11. Briem, G.J., Benediktsson, J.A., Sveinsson, J.R.: Multiple Classifiers Applied to Multisource Remote Sensing Data. IEEE Transactions on Geoscience and Remote Sensing (10), 2291–2299 (2002)
12. Bo, Y.C., Wang, J.F.: Combining Multiple Classifiers for Thematic Classification of Remotely Sensed Data. Journal of Remote Sensing (5), 555–564 (2005)
13. Zhang, X.Y., Feng, X.Z., Liu, W.: Urban Vegetation Categories Recognition by Multiple Classifier System from IKONOS Imagery. Journal of Southeast University (Natural Science Edition) (5), 399–403 (2007)
14. Foody, G.M., Boyd, D.S., Sanchez-Hernandez, C.: Mapping a specific class with an ensemble of classifiers. International Journal of Remote Sensing, 1733–1746 (2007)
15. Doan, H.T.X., Foody, G.M.: Increasing soft classification accuracy through the use of an ensemble of classifiers. International Journal of Remote Sensing (10), 4609–4623 (2007)
16. Dehghani, H., Ghassemian, H., Keshavarz, A.: A Multiple Classifier Template For Hyperspectral Images Classification. In: 2005 IEEE Geoscience and Remote Sensing Symposium, pp. 3776–3779 (2005)
17. Benediktsson, J.A., Chanussot, J., Fauvel, M.: Multiple classifiers in remote sensing: from basics to recent developments. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 501–512. Springer, Heidelberg (2007)
18. Briem, G.J., Benediktsson, J.A., Sveinsson, J.R.: Multiple Classifiers Applied to Multisource Remote Sensing Data. IEEE Transactions on Geoscience and Remote Sensing 40(10), 2291–2300 (2002)
19. Witten, I.H., Frank, E.: Data Mining Practical Machine Learning Tools and Techniques, 2nd edn. China Machine Press, Beijing (2006)
20. Hassiba, N., Youcef, C.: Multiple support vector machines for land cover change detection: An application for mapping urban extensions. ISPRS Journal of Photogrammetry & Remote Sensing (61), 125–133 (2006)
21. Liu, C.P., Dai, J.F., Zhong, W., et al.: Multi-source Remote Sensing Information Based on Fuzzy Evidence Theory. Pattern Recognition and Artificial Intelligence 6, 213–218 (2003)
22. Sun, Q., Ye, X.Q., Gu, W.K.: A New Combination Rules of Evidence Theory. Acta Electronica Sinica (8), 117–119 (2000)

Ensemble Strategies for Classifying Hyperspectral Remote Sensing Data

Xavier Ceamanos[1], Björn Waske[2], Jón Atli Benediktsson[2], Jocelyn Chanussot[1], and Johannes R. Sveinsson[2]

[1] GIPSA-LAB, Signal & Image Department, Grenoble Institute of Technology, INPG BP 46 - 38402 Saint Martin d'Hères, France
[2] University of Iceland, Faculty of Electrical and Computer Engineering, Hajararhagi 2-6, 107 Reykjavik, Iceland

Abstract. The classification of hyperspectral imagery, using multiple classifier systems is discussed and an SVM-based ensemble is introduced. The data set is separated into separate feature subsets using the correlation between the different spectral bands as a criterion. Afterwards, each source is classified separately by an SVM classifier. Finally, the different outputs are used as inputs for final decision fusion that is based on an additional SVM classifier. The results using the proposed strategy are compared to classification results achieved by a single SVM and other well known classifier ensembles, such as random forests, boosting and bagging.

Keywords: hyperspectral, land cover classification, support vector machines, multiple classifier systems, classifier ensmeble.

1 Introduction

Hyperspectral data provide detailed spectral information from land cover, ranging from the visible to the short-wave infrared region of the electromagnetic spectrum. Nevertheless the classification of hyperspectral imaging is challenging, due to the high-dimension of the data sets. Particularly with a limited number of training samples the classification accuracy (of conventional statistical classifiers) can be limited. Hughes [1] showed that with a limited number of training samples the classification accuracy decreases after a maximum is achieved. Thus, it requires sophisticated classification algorithms to use detailed hyperspectral information comprehensively. In several remote sensing studies it was demonstrated that Support Vector Machines (SVM) perform better than or at least comparable to other classifiers in terms of accuracy, even when applied to hyperspectral data sets [2],[3]. One reason for this success might be the underlying concept of SVM classifiers. Their aim is to discriminate two classes by constructing an optimal separating hyperplane to the training samples within a multi-dimensional feature space, by using only the closest training samples of each class [4]. Consequently, the approach only considers training data close to the class boundary and performs well with small training sets.

Multiple classifier systems (MCS) or classifier ensembles are another machine learning concept, which has been applied to remote sensing data sets recently [5]. By combining different independent classifiers, MCS can improve the classification accuracy in comparison to a single classifier. In [6]-[8] ensemble strategies were successfully applied to hyperspectral data sets. In [9] an SVM base classifier system was introduced, to classify multisensor imagery. Each data set, a multitemporal image and a set of multitemporal SAR data, was individually classified by SVM. Afterwards the outputs were fused by an additional SVM classifier. The results demonstrate that the proposed classification concept outperforms other parametric and non-parametric classification techniques (i.e., maximum likelihood classifier, decision tree, and boosted decision tree) including a single SVM. Moreover, the fusion step by an additional SVM classifier seems more efficient than other approaches, such as a simple majority vote. In [10] the simultaneous use of a neural network and a statistical method was discussed for classifying a hyperspectral data set. The image was separated into several feature subsets, using the correlation coefficient between the different bands. Afterwards each subset was individually classified by a statistical classifier and a neural network. To generate the final map the outputs were combined by decision fusion. In [11] a similar concept was used for economic forecasting. The feature space was separated into different subsets, using mutual information as criterion. Therefore, features within the same group are more similar to each other than compared to features belonging to different subsets. Whereas in [10] the individual feature subsets were used as input for the classifier ensemble, in [11] the input feature sets were generated by selecting individual features from each subset.

In regard to these results, it seems interesting to apply the approach introduced in [9] to a hyperspectral data set. To generate multiple sources, the original hyperspectral data are split into several smaller data sources, following the correlation between bands as proposed in [10]. The subsets are classified by an individual SVM classifier. Finally the different outputs are combined by an additional SVM classifier [9]. The results are compared to those achieved by a single SVM classifier using the whole hyperspectral data set as well as other ensemble methods, such as boosted decision trees and random forests.

The paper is organized as follows. The classification techniques are reviewed in Section 2, followed by the SVM ensemble in Section 3. The data set is introduced in Section 4. Experimental results are given in Section 5, and, finally, conclusions are discussed in Section 6.

2 Classifier Algorithms

2.1 Support Vector Machines

Support Vector Machines fit an optimal separating hyperplane to the training data of two classes in a multi-dimensional feature space. In linearly non-separable cases, the input space is mapped into a high dimensional feature space, using a so-called kernel function [4]. A detailed overview on the general concept of SVM is given in [12]. A brief introduction is given below: Let us assume that a training data set of ℓ samples, in a d-dimensional feature space $\Re^d$, is given by x_i with their corresponding class labels $y_i = \pm 1$, $\Omega = \{(\mathbf{x}_i, y_i) \mid i \in [1, \ell]\}$.

The linear hyperplane $f_l(x) = wx + b$ is given by the normal vector w and the bias b, with $|b| / \|w\|$ as the distance between the hyperplane and the origin, where $\|w\|$ is the Euclidean norm from w. The margin maximization results in the following optimization problem:

$$min \left[\frac{w^2}{2} + C \sum_{i=1}^{\ell} \zeta_i \right] \tag{1}$$

with ζ_i as slack variables and C as regularization parameter. The constant C penalizes training errors, i.e., training samples that are located on the wrong side of the hyperplane. The final SVM function for a non-linear separable case is described as follows:

$$f_n(x) = \left(\sum_{i=1}^{\ell} \alpha_i y_i k(x_i, x_j) + b \right) \tag{2}$$

where α_i are Lagrange multipliers.

Thanks to the *kernel-trick* it is possible to work within the newly transformed feature space, without knowing the explicit mapping, but only knowing the kernel function $k(x_i, x_j)$. In this study a common radial basis function (RBF) kernel was used:

$$k(x_i, x_j) = exp \left[-\gamma \|x_i - x_j\|^2 \right] . \tag{3}$$

The training of an SVM classifier requires the adequate definition of the kernel parameter γ and the regularization parameter C, which is usually done by a grid-search. Various combinations of C and γ are tested and the combination that yields the highest accuracy (based on a cross validation) is taken. A one-against-one rule was used, which generates a binary SVM for each possible pair-wise classification problem.

In contrast to other classifier algorithms, which provide probability measurements (e.g., Bayesian classifiers) and class labels (e.g., decision trees), respectively, the output image of a SVM classifier (Eq. 2) contains the distance of each pixel to the hyperplane of the binary classification problem. This information is used to determine the final result.

2.2 Multiple Classifier Systems

Multiple classifier systems combine variants of the same base classifier or different algorithms [13]. In doing so the total accuracy is usually increased, compared to the classification accuracy achieved by a single classifier [13]. Several different concepts have been introduced, which were also applied successfully to in diverse remote sensing studies [9],[10],[14],[15]. However two main techniques exist: *boosting* and *bagging*.

Boosting techniques, such as AdaBoost.M1 [16], are concepts to improve the performance of any (weak) classifier. During the initialization all training samples are equally weighted. The weights of the training samples are iteratively modified after each step and the next base classifier $\mathcal{C}_B$ within the system is trained on the reweighed samples. The weights of correctly classified training samples decreases, while misclassified samples are assigned a stronger weight. In doing so the classifier is focusing on "difficult" training samples. The approach is described as follows:

```
Input: A training set $\Omega = \{(\mathbf{x}_j, y_j)\}_{j=1}^{\ell}$, base classifier $\mathcal{C}_B$ and number
of classifiers $I$.
 1. $\Omega_1 = \Omega$ and weight$(x_j) = 1$ for $j = 1 \ldots l$ $(x \in S_1)$
 2. FOR $i = 1$ to $I\{$
 3.     $C_i = \mathcal{C}_B(\Omega_i)$
 4.     calculate error rate $\epsilon_i$
 5.     if $\epsilon_i > 0.5$, terminate procedure
 6.     calculate weight $\beta_i = \epsilon_i/(1 - \epsilon_i)$
 7.     for each $x_j \in \Omega_i\{$ if $C_i(x_j) \neq y_j$ then
            weight$(x_j)$ = weight$(x_j) \cdot \beta_i\}$.
 8.         normalize weights that the total sum of weights is 1$\}$.
 9. END
10. $C^*(x) = \arg\max \sum_{C_i(x)=y} \log(1/\beta_i)$
```

In contrast to this ***bagging*** (bootstrap aggregating) is based on resampling instead of re-weighting, thus it does not change the distribution of the training data and all classes are equally weighted [17]. Usually a random and uniform selection is performed, generating a training set with ℓ samples from a training set of same size ℓ. This random selection is performed *with replacement*, i.e., a sample can be selected several times in a set, whereas another sample is not considered in this particularly training set. Each individual training set is used to train the base classifier, thus, different outputs are generated. A simple majority vote is used to determine the final classification result. Bagging is described as follows:

```
Input: A training set $\Omega=\{(\mathbf{x}_j, y_j)\}_{j=1}^{\ell}$, the base classifier $\mathcal{C}_B$ and num-
ber of randomly generated training sets $I$.
 1. FOR $i = 1$ to $I\{$
 2.     $\Omega_i$ = training set from $\Omega$
 3.     $\Omega_i = \mathcal{C}(\Omega_i)\}$
 4. END
 5. the class with the maximum number of votes is chosen
```

The ***random forest*** (RF) method is a combination of bagging of the training samples as well as the attributes. RF is an ensemble of decision tree classifiers $DT(x, \theta_i), i = 1, ...,$ where θ_i are independent identically distributed random vectors and x is an input pattern [18]. Thus, each tree within the classifier systems ensemble is trained on a subset of the original training samples. In addition the feature subset is generated randomly at each split of a tree. RF are well suited for classifying high dimensional data sets, because the computational complexity is simplified by decreasing the number of input features (and training samples) at each node.

3 Data Sets

An AVIRIS (Airborne Visible InfraRed Imaging Spectrometer) data set was collected on a cloud-free day, surrounding the region of the volcano Hekla in South Iceland. The sensor operates from the visible to mid-infrared region of the electromagnetic spectrum (i.e., 0.4μm to 2.4μm). The system has four spectrometers and 224 data channels. Because spectrometer 4 was not working properly during the image acquisition, 64 bands (1.84μm to 2.4μm) were deleted from the imagery, as the first channels for all the other spectrometers, but those channels were blank. Thus, the image used in this study contains 157 bands. The data set covers 2048 x 614 pixels, with a spatial resolution of 20 m [10]. Twenty-four land cover classes were considered during the classification, mainly lava flows from different eruptions, partly covered by vegetation. The available ground truth information was equally split into independent training and test data. Moreover, different training data sets were generated, containing 25 and 100 samples per class, as well as a set with all available training data (total: 17491). The validation set contains 16667 samples.

4 SVM Ensemble Strategy

The proposed SVM classifier ensemble is based on the application of SVM on different data sources and a fusion of the outputs by an additional SVM. To generate different sources, the hyperspectral image is separated into feature subsets, in accordance to the correlation matrix (see Fig. 2). The elements in the correlation matrix Σ are defined by the absolute correlation r between the spectral response of individual bands s_i and s_j in a d-dimensional feature space is defined as:

$$r_{S_iS_j} = \left| \frac{d\sum_1^d s_is_j - \sum_1^d s_i \sum_1^d s_j}{\sqrt{\left[d\sum_1^d s_i^2 - \left(\sum_1^d s_i\right)^2\right]\left[d\sum_1^d s_j^2 - \left(\sum_1^d s_j\right)^2\right]}} \right| \quad (4)$$

Figure 2 shows the correlation matrix for the data set. Blue regions show a low correlation, whereas a high correlation is indicated by red. A visual interpretation of the matrix points out three main regions of high correlation, ranging from band 3 to 29, 30 to 105 and 111 to 150. The low correlation values in the remaining bands, indicates noise (e.g., water absorption) and thus, the bands

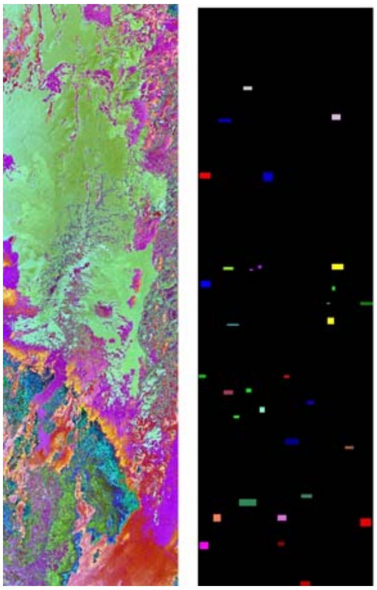


Fig. 1. AVIRIS data set, sourrounding the region of the volcano Hekla, South Iceland and coressponding test data with 24 classes

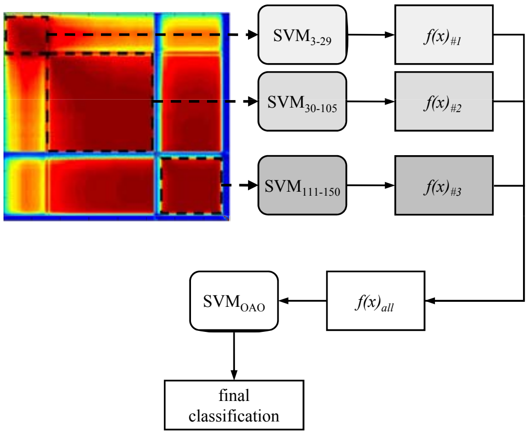


Fig. 2. Schematic overview on the SVM-based ensemble (after Waske and Benediktsson, 2007). The visual interpretation of the correlation matrix points out three different major regions, which are used for the application of the proposed SVM ensemble.

are removed. The remaining 143-band data set was used for the classifcation by SVM, Boosting, RF, etc.. After generating three feature subsets, individual SVM classifiers were applied to each subset. The outputs were fused by a final SVM (see Fig. 2).

5 Experimental Results

The SVM were trained on the different feature subsets and the whole image. The training and parameter selection was performed using LIBSVM in a MATLAB environment [20]. The best values for γ and C were selected in an user defined range of possible parameters based on a leave-one-out cross validation. The outputs generated for the three feature subsets (see Fig.2) were then used for the decision fusion process, which was based on the application of another SVM (see Fig.2). In addition to the SVM, three different ensemble strategies were applied on the data sets, boosting, bagging and random forests (RF), with varying ensemble sizes (i.e., 10, 25, 50, 100). Boosting (i.e., AdaBoost.M1) and bagging were performed with a j4.8 decision tree, which is an implementation of the well-known C4.5 decision tree. All three ensembles were applied by using the WEKA data mining software [19].

In Table 1 the classification accuracies for the different results are given. The results of the proposed SVM classification show that the total accuracy is increased between 2.3% and 4.6% when compared to a single SVM classifier. Comparing the different DT-based classifier systems, it can be assessed that boosting and random forests achieve significantly higher accuracies compared to bagging, which even performs less accurate than a single SVM does. Boosting and RF achieve very similar results. Whereas boosting perform slightly better with large training data sets, the latter approach is more adequate for a smaller training set size. All three DT-based ensemble methods show a typical increase in the classification accuracy with an increasing number of classifiers within the ensemble (not presented in detail). Thus, in the following discussion only the results achieved by 100 iterations are considered.

The accuracy assessment demonstrates that the proposed SVM ensemble strategy can outperform boosting and RF in terms of accuracy, depending on the number of available training samples. With a small number of samples (i.e., 25 per class) DT-based boosting and RF yield higher classification accuracy (i.e., 73.2% and 74.3%) than the proposed method (71.4%). Although a larger training set size results in an increase of the classification accuracies for all methods, the increase is most significant for the SVM ensemble. The classifications that

Table 1. Overall test accuracy in percentage, using the proposed SVM-based classifier system and other ensemble methods (with 100 iterations) with different training sample sets (25 samples per class (TR 25), 100 samples per class (TR 100) and all training samples)

	Overall test accuracy [%]		
Method	**TR 25**	**TR 100**	**all training samples**
single SVM	66.8	79.4	90
SVM ensemble	71.4	82.2	92.3
Boosting	73.2	82.9	89.2
Bagging	65.3	76.1	85.4
RF	74.3	82.9	88.1

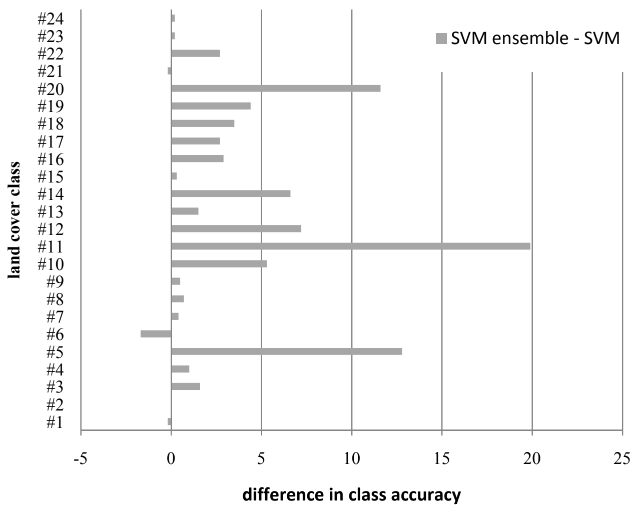


Fig. 3. Differences in class accuracies, achived by the SVM ensemble as compared to a standard SVM

are based on the medium training set size show very similar accuracies. However, using the largest training set with all available samples, the proposed ensemble strategy results in accuracy of 92.3%, which is approximately 3% to 4% higher compared to the results achieved by boosting and RF.

The good performance of the proposed classifier ensemble is also underlined by the class accuracy. Comparing the class accuracies, achieved by the SVM-ensemble with those by a standard SVM classifier, it can be observed that the proposed strategy achieved the higher class accuracies in most cases. In Figure 3 the differences between the accuracies for different land cover classes are shown. The differences significantly tends towards the positive, i.e, the proposed strategy outperforms a single SVM in terms of the class accuracy, in most cases.

6 Discussion and Conclusion

In this paper, the problem of classifying hyperspectral imagery was addressed. A multiple classifier system was proposed, which is based on the fusion of SVM. The classification strategy is based on the combination of different SVM classifiers that are applied to several subsets within the feature space. To generate different subsets, the whole data set was separated, using the correlation between spectral bands. Besides the proposed strategy a single SVM and different well-known classifier ensembles were applied on the data set. Their performance was compared to the proposed method, varying the number of training samples.

Experimental results show that the proposed SVM fusion outperforms an SVM classifier in terms of total accuracy, irrespectively of the number of available training samples. In contrast to this, other ensembles such as boosting and random forests can outperform the proposed strategy in terms of accuracy, particularly with a small number of training samples. Nevertheless the SVM ensemble is interesting, when a larger number of training samples is available. In this case it performs better than boosting, bagging and random forests in terms of accuracy. Overall, the proposed method seems to be a promising, alternative classification strategy for hyperspectral remote sensing data and can yield higher accuracies than other well-known ensemble methods. In our future research the computational complexity needs to be reduced. Moreover the impact of the subset generation (e.g., number of subsets) on the overall accuracy will be investigated.

References

1. Hughes, G.F.: On the mean accuracy of statistical pattern recognizers. IEEE Trans. on Information Theory 14, 55–63 (1968)
2. Melgani, F., Bruzzone, L.: Classification of hyperspectral remote sensing images with support vector machines. IEEE Trans. Geosci. and Remote Sens. 42, 1778–1790 (2004)
3. Pal, M., Mather, P.M.: Some issues in the classification of DAIS hyperspectral data. Int. J. Remote Sens. 27, 2896–2916 (2006)
4. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
5. Benediktsson, J.A., Chanussot, J., Fauvel, M.: Multiple Classifier Systems in Remote Sensing: From Basics to Recent Developments. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 501–512. Springer, Heidelberg (2007)
6. Ham, J., Chen, Y.C., Crawford, M.M., Ghosh, J.: Investigation of the random forest framework for classification of hyperspectral data. IEEE Trans. on Geosci. and Remote Sens. 43, 492–501 (2005)
7. Joelsson, S.R., Benediktsson, J.A., Sveinsson, J.R.: Random forest classification of remote sensing data. In: Chen, C.H. (ed.) Signal and Image Processing for Remote Sensing, pp. 327–344. CRC Press, Boca Raton (2007)
8. Cheung-Wai Chan, J., Paelinckx, D.: Evaluation of Random Forest and Adaboost tree-based ensemble classification and spectral band selection for ecotope mapping using airborne hyperspectral imagery. Remote Sens. of Env. 112, 2999–3011 (2008)
9. Waske, B., Benediktsson, J.A.: Fusion of Support Vector Machines for Classification of Multisensor Data. IEEE Trans. Geosci. and Remote Sens. 45, 3858–3866 (2007)
10. Benediktsson, J.A., Kanellopoulos, I.: Classification of Multisource and Hyperspectral Data Based on Decision Fusion. IEEE Trans. Geosci. and Remote Sens. 37, 1367–1377 (1999)
11. Liao, Y., Moody, J.: Constructing heterogeneous committees using input feature grouping. In: Solla, S.A., Leen, T.K., Muller, K.-R. (eds.) Advances in Neural Information Processing Systems (NIPS) Conference, 1999, Denver, USA, vol. 12. MIT Press, Cambridge (2000)
12. Burges, C.J.C.: A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery 2, 121–167 (1998)
13. Polikar, R.: Ensemble Based Systems in Decision Making. IEEE Circuits and Systems Magazine 6, 21–45 (2006)

14. Briem, G.J., Benediktsson, J.A., Sveinsson, J.R.: Multiple Classifiers Applied to Multisource Remote Sensing Data. IEEE Trans. Geosci. Remote Sens. 40, 2291–2299 (2002)
15. Waske, B., van der Linden, S.: Classifying multilevel imagery from SAR and optical sensors by decision fusion. IEEE Trans. on Geosci. and Remote Sens. 46, 1457–1466 (2008)
16. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: 13th International Conference of Machine Learning, Bari, Italy (1996)
17. Breiman, L.: Bagging predictors. Mach. Learning 24, 123–140 (1996)
18. Breiman, L.: Random forests. Mach. Learning 45, 5–32 (2001)
19. Witten, I.H., Eibe, F.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
20. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm

Optimal Mean-Precision Classifier

David M.J. Tax, Marco Loog, and Robert P.W. Duin

Information and Communication Theory Group
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
D.M.J.Tax@tudelft.nl

Abstract. For pattern recognition problems where a small set of relevant objects should be retrieved from a (very) large set of irrelevant objects, standard evaluation criteria are often insufficient. For these situations often the precision-recall curve is used. An often-employed scalar measure derived from this curve is the mean precision, that estimates the average precision over all values of the recall. This performance measure, however, is designed to be non-symmetric in the two classes and it appears not very simple to optimize. This paper presents a classifier that approximately maximizes the mean precision by a collection of simple linear classifiers.

Keywords: Pattern recognition, performance evaluation, information retrieval, precision-recall graph.

1 Introduction

The standard performance measure in pattern recognition is the classification performance. In most real world applications the classification error is not well suited. For classification problems where classes are very imbalanced, or where the misclassification costs for different classes vary widely, the classification error can give a very unfair impression of the true performance. For the situations where the true misclassification costs are unknown, often the Receiver Operating Characteristic curve (ROC curve) is used. The ROC curve insensitive to class priors [Fla03], and the Area under the ROC curve (AUC) is often used as a scalar performance measure to compare classifiers [Bra97]. Classifiers are especially developed to directly optimize the AUC [BS05,FFHO02].

In the field of Information Retrieval (IR) one often does not use the ROC curve, but the Precision-Recall-curve. There the user queries a database of positive and negative documents, and retrieves the M most promising documents [SM83]. The performance measures therefore should incorporate the fact that just a limited number of objects is presented to the user. Two classical measures for Information Retrieval systems are the Precision and the Recall. Roughly speaking, the Precision measures how 'pure' the retrieved M documents are (so that just a few irrelevant documents are retrieved), while Recall measures how many of the total relevant documents are retrieved.

To evaluate a classifier, the number of retrieved documents M is often fixed, and the Precision and Recall are measured. Comparing two classifiers now involves comparing the two pairs of performances. To combine these two performances often the F-measure, which is the harmonic mean between the precision and recall, is used [vR79]. The comparison between two systems becomes harder when the operating point for the systems is not known beforehand, and the systems should work with varying operating points. In these situations it may be preferable to consider the mean precision, which is the precision averaged over a range of recall values.

Unfortunately, we don't know classifiers that directly optimize this mean precision. Often classifiers are constructed that minimize a classification error, or are tuned for a specific operating point in the precision-recall graph. This paper presents an algorithm that approximates the optimal mean precision by a combination of linear classifiers. In section 2 the mean precision is defined and rewritten such that it can estimated on a training dataset of objects. The estimator is decomposed in a sum of terms, where each term combines objects with equal precision. In section 3 a classifier is constructed that optimizes each term in the sum. In section 4 experiments are performed and in section 5 conclusions and further research directions are given.

2 Mean Precision

We have two classes, the positive target class, and the negative outlier or background class. It is assumed that the positive class has to be retrieved and that this class is (much) smaller than the negative class. Assume we have N^+ positive and N^- negative objects in our dataset: $x_j^+, j = 1, ..., N^+$ and $x_j^-, j = 1, ..., N^-$.

A classifier should rank all the objects in a dataset. Objects that are ranked below a threshold θ are 'accepted', otherwise they are 'rejected'. A graphical representation is given in Figure 1. The number of positive objects that is

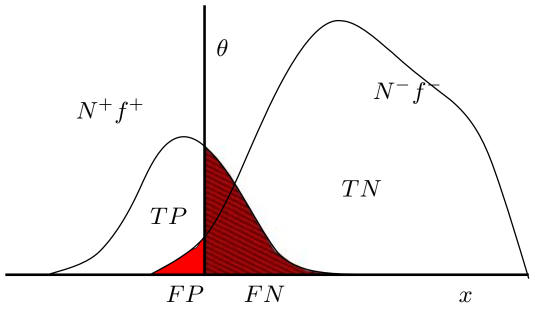


Fig. 1. Graphical representation of the distribution of the positive and negative class on the classifier output. Objects below the threshold θ are classified as positive. The different sets of data are shown: True Positive, True Negative, False Positive and False Negative.

accepted, is called the True Positives TP. The number of positive objects that is rejected, is called the False Positives FP.

The two performance measures, precision and recall, are defined as:

$$\text{precision}(\theta) = \frac{TP(\theta)}{TP(\theta) + FP(\theta)} \tag{1}$$

$$\text{recall}(\theta) = \frac{TP(\theta)}{TP(\theta) + FN(\theta)} = \frac{TP(\theta)}{N^+} \tag{2}$$

2.1 Mean Precision for Given 1D Distributions

Consider a one dimensional feature x for which the true distributions of the positive and negative classes f^+ and f^- are known. Then the cumulative distributions F^+ and F^- are defined, and the total cumulative sum becomes:

$$F(x) = N^+F^+(x) + N^-F^-(x). \tag{3}$$

The True Positives and False Positives can be written as:

$$TP(\theta) = N^+F^+(\theta), \quad FP(\theta) = N^-F^-(\theta). \tag{4}$$

The recall for a given threshold θ becomes:

$$\text{recall}(\theta) = \frac{N^+F^+(\theta)}{N^+} = \int_{-\infty}^{\theta} f^+(u)du \tag{5}$$

and the precision:

$$\text{precision}(\theta) = \frac{N^+F^+(\theta)}{F(\theta)} = \frac{N^+ \int_{-\infty}^{\theta} f^+(u)du}{N^+ \int_{-\infty}^{\theta} f^+(u)du + N^- \int_{-\infty}^{\theta} f^-(u)du}. \tag{6}$$

The mean precision is now defined as the precision averaged over all values of the recall. This can be written as an average over θ when a coordinate transform is applied. Note that

$$\frac{d\text{recall}(\theta)}{d\theta} = \frac{dF^+(\theta)}{d\theta} = f^+(\theta). \tag{7}$$

Therefore the integration variable $d\text{recall}(\theta)$ can be replaced by $f^+(\theta)d\theta$:

$$\overline{\text{prec}} = \int_{-\infty}^{\infty} \frac{N^+F^+(\theta)}{F(\theta)} f^+(\theta)d\theta = \int_{-\infty}^{\infty} \frac{N^+ \int_{-\infty}^{\theta} f^+(u)du}{N^+ \int_{-\infty}^{\theta} f^+(u)du + N^- \int_{-\infty}^{\theta} f^-(u)du} f^+(\theta)d\theta. \tag{8}$$

2.2 Mean Precision Using Sampled Distributions

When the exact distributions f^+ and f^- are not available, the distributions can be approximated using sampled versions. Assume that the objects $x_j^+, j = 1, ..., N^+$ are samples from the positive class, and $x_j^-, j = 1, ..., N^-$ are from the negative class. For simplicity later, we assume that the positive and negative objects are ordered: $x_1^+ < x_2^+ < ..., < x_{N^+}^+$. When no superscript is given, the data can be of any of the two classes $x_j, j = 1, ..., N$, where $N = N^+ + N^-$. Then:

$$f^+(x) = \sum_{j=1}^{N^+} \delta(x - x_j^+), \quad F^+(x) = \sum_{j=1}^{N^+} \mathcal{I}(x \geq x_j^+), \tag{9}$$

where $\delta(x)$ is a Dirac-Delta function and $\mathcal{I}(.)$ is the indicator function $\mathcal{I}(A) = 1$ if the statement A is true and $\mathcal{I}(A) = 0$ otherwise).

Substituting this in (5) and (6) gives for a single value of θ:

$$\text{precision}(\theta) = \frac{\sum_{j=1}^{N^+} \mathcal{I}(\theta \geq x_j^+)}{\sum_j^N \mathcal{I}(\theta \geq x_j)}, \qquad \text{recall}(\theta) = \sum_{j=1}^{N^+} \mathcal{I}(\theta \geq x_j^+). \tag{10}$$

To find the mean precision, we have to average over all values of recall. The average over all positive objects x_i is computed by:

$$\overline{\text{prec}} = \frac{1}{N^+} \sum_{i=1}^{N^+} \text{prec}(x_i^+) = \frac{1}{N^+} \sum_{i=1}^{N^+} \frac{\sum_{j=1}^{N^+} \mathcal{I}(x_i^+ \geq x_j^+)}{\sum_{j=1}^{N} \mathcal{I}(x_i^+ \geq x_j)} = \frac{1}{N^+} \sum_{i=1}^{N^+} \frac{i}{\sum_{j=1}^{N} \mathcal{I}(x_i^+ \geq x_j)}. \tag{11}$$

Note that in the last step we assumed that the objects $x_1^+, x_2^+, ..., x_{N^+}^+$ are ordered ($x_1^+ < x_2^+ < ..., < x_{N^+}^+$), such that the sum in the numerator can be reduced to: $\sum_{j=1}^{N^+} \mathcal{I}(x_i^+ \geq x_j^+) = i$.

2.3 Decomposition of the Mean Precision

The sum given in the denominator of (11) can be decomposed into two parts; one sum over all positive objects and one over all negative objects. This results in:

$$\sum_{j=1}^{N} \mathcal{I}(x_i^+ \geq x_j) = \sum_{j=1}^{N^+} \mathcal{I}(x_i^+ \geq x_j^+) + \sum_{j=1}^{N^-} \mathcal{I}(x_i^+ \geq x_j^-) = i + c_i^-. \tag{12}$$

We define S_0 as the subset of the positive objects x_i^+, $i = 1, ..., m_0$ for which $c_i^- = \sum_{j=1}^{N^-} \mathcal{I}(x_i^+ \geq x_j^-) = 0$. These are the objects that do not have any negative objects with a lower feature value. In figure 2 these are the three positive objects that are located left of 0.1. For these objects the terms in the sum in (11) becomes one. Next we can define the sets S_k as the subsets of positive objects for which $c_i^- = k$, and define the cardinalities of the sets $m_k = |S_k|$:

$$S_k = \left\{ x_i^+ \middle| \sum_{j=1}^{N^-} \mathcal{I}(x_i^+ \geq x_j^-) = k \right\}, \quad k = 0, .., N^-. \tag{13}$$

For instance, the set S_1 in figure 2 contains the positive objects that have just a single negative object left of them, so that are all positive objects between 0.1 and 1.0. For this example there is two objects (around $x = 0.5$), and therefore $m_1 = 2$. Note also that $m_6 = m_7 = ... = m_{12} = 0$.

The total mean precision can now be written as (combining (11) and (12)):

$$\begin{aligned}\overline{\text{prec}} &= \frac{1}{N^+} \sum_{i=1}^{N^+} \frac{i}{i + \sum_{j=1}^{N^-} \mathcal{I}(x_i^+ \geq x_j^-)} \qquad (14)\\ &= \frac{1}{N^+} \sum_{i=1}^{N^+} \frac{i}{i + c_i^-} = \frac{1}{N^+} \left[\sum_{i=1}^{m_0} \frac{i}{i} + \sum_{i=m_0+1}^{m_0+m_1} \frac{i}{i+1} + \sum_{i=m_0+m_1+1}^{m_0+m_1+m_2} \frac{i}{i+2} + ... \right]\end{aligned}$$

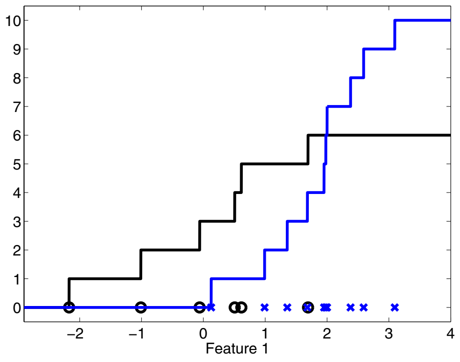


Fig. 2. The (unnormalized) cumulative distributions $N^+F^+(x)$ and $N^-F^-(x)$ for the positive objects (small circles) and the negative objects (crosses)

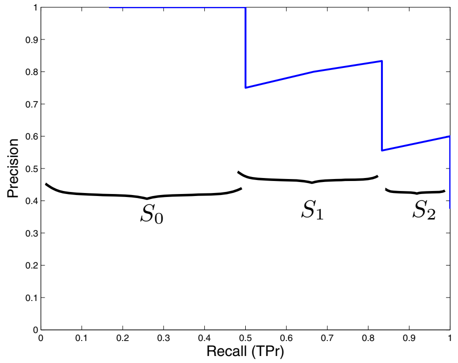


Fig. 3. The resulting precision-recall graph for the data that is shown in Figure 2

In figure 3 the Precision-Recall curve for the data shown in figure 2 is shown. The different subsets S_k of positive objects define different ranges in the recall. For this example only three subsets, S_0, S_1 and S_2 are non-empty, and they are indicated in the graph. The first three positive objects on the left are element of S_0 and have a precision of 1. The two objects in S_1 have two different precision values of 4/5 and 5/6.

3 Optimizing Mean Precision

For a given feature x the mean precision (11) or (14) can directly be computed. Assume we would like to define a new feature as a linear combination of original features:

$$z = \mathbf{w}^T\mathbf{x} \tag{15}$$

in such a way that it optimizes the mean precision:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} \frac{1}{N^+} \sum_{i=1}^{N^+} \frac{i}{i + \sum_{j=1}^{N^-} \mathcal{I}(\mathbf{w}^T\mathbf{x}_i^+ \geq \mathbf{w}^T\mathbf{x}_j^-)} \tag{16}$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}} \frac{1}{N^+} \left[\sum_{i=1}^{m_0} \frac{i}{i} + \sum_{i=m_0+1}^{m_0+m_1} \frac{i}{i+1} + \sum_{i=m_0+m_1+1}^{m_0+m_1+m_2} \frac{i}{i+2} + \ldots \right] . \tag{17}$$

Unfortunately, this is a complicated nonlinear optimization problem. When the weights $\mathbf{w}$ are changed a bit, the ordering of the objects may change drastically, and objects move between the different sets S_k. This causes the numbers m_k to change and therefore also the sums in (17). To optimize all terms simultaneously appears to be very hard (we could not reduce it to a easily-solvable optimization problem). Therefore we decide to optimize (17) term by term in a greedy optimization.

3.1 Optimizing the First Term in Mean Precision

To optimizing the first term in (17) we try to find that direction $\mathbf{w}$ in feature space such that the maximum number of positive objects do not have a single negative object below them. That means that we try to find a classifier $f(\mathbf{x}) = \mathcal{I}(\mathbf{w}^T\mathbf{x} < \theta)$ that maximizes m_0 first. This can be done by the following (linear programming) optimization procedure:

$$\min_{\mathbf{w}} |\mathbf{w}| + C \textstyle\sum_{i \in +} \xi_i \tag{18}$$

$$\text{s.t.} \qquad \mathbf{w}^T\mathbf{x}_i^- \geq \theta, \qquad \text{for negative objects} \tag{19}$$

$$\mathbf{w}^T\mathbf{x}_i^+ \leq \theta + \xi_i, \quad \xi_i \geq 0 \qquad \text{for positive objects.} \tag{20}$$

In the constraints (19) all the negative objects x_i^- are forced to be above the threshold θ, so the negative objects are classified without error and the term $\sum_{j=1}^{N^-} \mathcal{I}(\mathbf{w}^T\mathbf{x}_i^+ \geq \mathbf{w}^T\mathbf{x}_j^-)$ in (16) becomes 0. Positive objects x_i^+ that are not on the correct side (i.e. below) of the decision boundary are punished by a so-called slack ξ_i. This is defined in the first constraint (20). The sum of the slacks is minimized in the function (18), together with a regularization term on the L_1-norm of the weight vector $\mathbf{w}$.

This optimization procedure finds a linear combination of original features, or a linear classifier, such that the number of correctly classified positive objects is as large as possible, and such that none of the negative objects is misclassified.

3.2 Optimizing the Next Terms in Mean Precision

The previous formulation only considers the first term in (17). To maximize the second term in (17), or to find the largest set S_1, we have to find the largest set of positive objects that have *one* negative object below them. This optimization becomes very complicated. Therefore we decided to remove the negative object that is the closest to the set S_0. That is the object for which the constraint (20) is violated the first[1]. After this object is removed from the training set, the optimization (18) is run again, selecting the set S_1.

This process is repeated until all positive objects have been classified to the positive class, or until a pre-specified value for the recall is obtained. Assume that a certain recall r is required. The number of classifiers n that has to be trained to obtain at least this recall, becomes $n : \sum_{i=1}^{n} m_i \geq N^+ r$. This depends on the number of positive objects m_k in each of the sets S_k, which again depends on the data. It is therefore not clear beforehand how many classifiers have to be trained to obtain a certain recall.

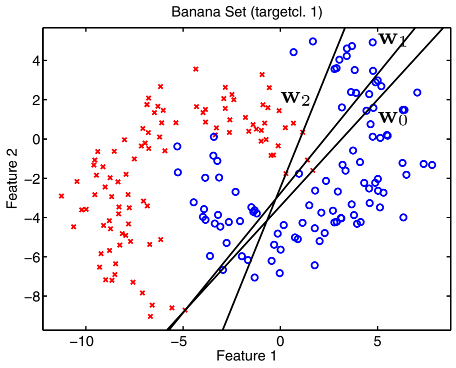


Fig. 4. Scatterplot with a positive and negative class (circles and crosses respectively), containing the sequence of classifiers $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_3$

[1] In linear programming optimization algorithms each of the constraints obtains a dual variable that indicates how heavily this constraint has to be enforced. The object corresponding to the constraint with the highest dual variable is selected to be removed.

This process is also shown in figure 4. The first classifier $\mathbf{w}_0$ separates the positive objects (indicated by the circles) from the negative objects (crosses). None of the negative objects are misclassified, and around 50% of the positive objects are correct. In the second iteration the rightmost negative object (at position $(1.6, -1.6)$) is removed, and the new classifier $\mathbf{w}_2$ is trained.

3.3 Combining the Classifiers

Unfortunately, the procedure does not result in a single unique $\mathbf{w}$, but in a collection of classifiers $\mathbf{w}_k$. When a new object has to be classified, the classifiers have to be combined into a single output, depending on the actual operating point that is chosen.

Following the philosophy of optimizing (17) term by term, an iterative scheme has to be used. First classify the new object $\mathbf{w}$ by $\mathbf{w}_0$. When $\mathbf{z}$ is classified positive by $\mathbf{w}_0$, the classification is finished and the object $\mathbf{z}$ is labeled positive. When $\mathbf{w}_0$ classifies $\mathbf{z}$ as negative, the object is presented to the next classifier $\mathbf{w}_1$, and the process repeats itself. The classifier labels the object as positive, or it moves it to the next classifier, until the last n-th classifier is reached, and the object is classified negative.

```
Input: Datasets $\{x_j^+, j = 1, ..., N^+\}$ and $\{x_j^-, j = 1, ..., N^-\}$, M
Output: Classifier $\mathbf{w}$
for $k \leftarrow 0$ to $M$ do
    optimize $\mathbf{w}_k$ using (17);
    find object $\mathbf{x}^* \in \{x_j^-\}$ with the largest dual variable;
    remove object $\mathbf{x}^*$ from the negative examples;
end
combine classifiers $\mathbf{w}_k$ to classifier $\mathbf{w}$
```

Algorithm 1. The optimal mean-precision algorithm

In algorithm 1 shows a high-level overview of the steps that are taken in the (approximate) optimization of the mean precision. It is not clear if this is the optimal approach. It is expected that the final classifier becomes more stable and robust when the individual classifiers $\mathbf{w}_k$ are averaged, but at the expense of the flexibility of the classifier. This is subject for further research.

4 Experiments

In this section we perform some experiments on some real world datasets to show the feasibility of the approach. The number of classifiers that is iteratively trained is also limited. Three versions are tested, using a single linear classifier (just $\mathbf{w}_0$, $M = 1$), using $M = 10$ classifiers and $M = 25$ classifiers. It was observed that for more than $M = 25$ classifiers the performance rarely improved.

We compare the Optimal-Mean-Precision formulation (OptPrec) with a collection of simple classifiers: the Linear Discriminant Analysis (LDA) and the Quadratic Discriminant (QD) [DHS01], the Logistic classifier [And82], the Parzen density classifier [Par62] and the Support Vector Classifier [Vap98]. For the Parzen classifier, the width parameter is optimized by maximizing the likelihood using leave-one-out on the training set. For the support vector classifier a linear kernel is used, where the C parameter is optimized using 10-fold cross-validation. In the experiments can show some results on the Imports85 dataset (159 objects in 25D), the Glass dataset (214 objects in 9D, where the classification task is to distinguish class 1 from the rest), and the Sonar dataset (208 objects in 60D). These datasets are taken from the UCI repository [NHBM98].

Table 1. The mean-precision (×100) for different classifiers and datasets. The best performances are indicated in bold. Results averaged over 10-fold stratified cross-validation. Values between the brackets indicate the standard deviation.

classifier	Imports85	Glass	Sonar
LDA	**85.2 (15.2)**	69.8 (11.7)	74.1 (15.6)
QD	**77.2 (18.0)**	**81.5 (19.0)**	77.1 (13.9)
Logistic	72.9 (23.2)	74.3 (15.1)	79.6 (14.9)
Parzen	**85.4 (17.3)**	75.4 (16.7)	80.9 (12.7)
SVM	**88.9 (14.4)**	72.1 (15.6)	77.9 (15.3)
OptPrec M=1	**90.8 (12.0)**	**79.1 (28.8)**	77.5 (17.2)
OptPrec M=10	**90.6 (12.1)**	**93.1 (9.4)**	76.6 (18.0)
OptPrec M=25	**89.4 (12.0)**	**93.1 (9.4)**	76.4 (17.8)
Kernel OptPrec M=1	74.6 (15.8)	80.7 (14.1)	**99.0 (1.0)**

The experimental results are shown in Table 1. All experiments are performed using 10-fold stratified cross-validation, and the performance measure is mean-precision. For each dataset the best average performance is written in bold. Furthermore, all performances that are not significantly worse (in terms of a one-sided t-test with a 5% significance level) are also written in bold.

The results on the Imports85 dataset show a common outcome: many classifiers have a similar performance, but the OptPrec slightly outperforms the other classifiers (although not significantly). Note also that the outcomes of the OptPrec are a bit more robust than of the other classifiers. The Glass dataset results show a situation where the OptPrec significantly outperforms the other approaches. Only the classifier that only maximizes the size of the first set m_0, is too unstable and gives poor results. The outcomes on the Sonar dataset shows that the OptPrec classifier is not always optimal in its linear implementation, but that for some datasets nonlinear decision boundaries are needed. This is implemented by kernelizing the linear classifier; the original data is mapped into a new feature representation using (in this case) the RBF kernel with an optimized width parameter σ. For the Sonar dataset it resulted in an almost perfect mean precision.

For many datasets the OptPrec classifier is not superior: in the cases where data is (almost) separable, or when the decision boundary is very nonlinear other classifiers may perform equally well, or better. Even more importantly, in small sample size classification problems it is often advantageous to use *all* the available training data for estimating the class conditional probabilities. It is hard to estimate the subset of positive objects in S_0 from a small training set. In these cases it might be advantageous to make the individual estimates $\mathbf{w}_k$ more robust and combine them. This is still an issue for further research.

5 Conclusions

This paper presents the derivation of a classifier that (approximately) optimizes the mean precision for a two-class classification problem. The classifier iteratively separates a part of the positive class from the negative class, such that the positive part is as 'pure' as possible (i.e. it does not contain any negative objects) and as large as possible. For each separation of a pure part, a classifier is obtained. When these classifiers are combined into a final classifier, the mean precision is optimized. Experiments show that for some datasets very good performances can be obtained. Further research is needed to investigate the possibility to optimize the mean precision in one step, how the classifiers have to be combined (in particular in the low sample size situation) and how it will perform on real world retrieval problems.

References

[And82] Anderson, J.A.: Logistic discrimination. In: Kirshnaiah, P.R., Kanal, L.N. (eds.) Classification, Pattern Recognition and Reduction of Dimensionality. Handbook of Statistics, vol. 2, pp. 169–191. North Holland, Amsterdam (1982)

[Bra97] Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159 (1997)

[BS05] Brefeld, U., Scheffer, T.: AUC miximizing support vector learning. In: Proceedings of ICML 2005 workshop on ROC analysis in Machine Learning (2005)

[DHS01] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley & Sons, Chichester (2001)

[FFHO02] Ferri, C., Flach, P., Hernandez-Orallo, J.: Learning decision trees using the area under the ROC curve. In: Proceedings of the ICML (2002)

[Fla03] Flach, P.: The geometry of ROC space: understanding machine learning metrics through ROC isometrics. In: Proceedings of the international conference on Machine learning 2003, pp. 194–201 (2003)

[NHBM98] Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998)

[Par62] Parzen, E.: On estimation of a probability density function and mode. Annals of Mathenatical Statistics 33, 1065–1076 (1962)

[SM83] Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)

[Vap98] Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)

[vR79] van Rijsbergen, C.J.: Information Retrieval, 2nd edn. Butterwort (1979)

A Multiple Expert Approach to the Class Imbalance Problem Using Inverse Random under Sampling

Muhammad Atif Tahir, Josef Kittler, Krystian Mikolajczyk, and Fei Yan

Centre for Vision, Speech and Signal Processing
University of Surrey
Guildford, GU2 7XH, UK
{m.tahir,j.kittler,k.mikolajczyk,f.yan}@surrey.ac.uk

Abstract. In this paper, a novel inverse random under sampling (IRUS) method is proposed for class imbalance problem. The main idea is to severely under sample the negative class (majority class), thus creating a large number of distinct negative training sets. For each training set we then find a linear discriminant which separates the positive class from the negative class. By combining the multiple designs through voting, we construct a composite between the positive class and the negative class. The proposed methodology is applied on 11 UCI data sets and experimental results indicate a significant increase in Area Under Curve (AUC) when compared with many existing class-imbalance learning methods.

1 Introduction

Many real world classification problems are represented by highly imbalance data sets, that is, the number of samples from one class is much smaller than from another. This is known as class imbalance problem and is often reported as an obstacle to construct a model that can successfully discriminate minority samples from majority samples. Generally, the problem of imbalanced data sets occurs when one class represents a rare or uncommon concept while the other class represents the anti-concept, so that the examples from the anti-concept class outnumber the examples from the concept class. This type of data is found, for example, in the image retrieval concept detection problem where only few images belong to the concept class; in medical record databases for rare diseases where a small number of patients would have a particular disease.

There is a great deal of research on learning from imbalanced data sets reported in the literature [1,8,6]. The most commonly used methods to handle imbalanced data sets involve under sampling or over sampling of the original data set. Over sampling aims to balance class populations through replicating the minority class examples while under sampling aims to balance the class populations through the elimination of majority class examples.

In this paper, a novel inverse random under sampling (IRUS) method is proposed for the class imbalance problem in which the ratio of the respective training set cardinalities is inversed. The idea is to severely under sample the negative class (majority class), thus creating a large number of distinct negative training sets. For each training set we then find a linear discriminant which separates the positive class from the negative samples. As the number of positive samples in each training set is greater than the number of negative samples, the focus in machine learning is on the positive class and consequently it can invariably be successfully separated from the negative training samples. Thus each training set yields one classifier design. By combining the multiple designs through voting, we construct a composite between the positive class and the negative class. We shall argue that this boundary has the capacity to delineate the positive class more effectively than the solutions obtained by conventional learning. We shall show experimentally on standard benchmarking data that the proposed method leads to significant improvements in performance.

This paper is organized as follows. Section 2 provides briefly review several class imbalance methods followed by proposed inverse random under sampling method (IRUS) in section 3. Section 4 describes the experimental setup followed by results and discussion in Section 5. The paper is drawn to conclusion in Section 6.

2 Related Work

As discussed in Section 1, the most commonly used methods to handle imbalanced data sets involve under sampling or over sampling of the original data sets. Random over sampling and random under sampling are the most popular non-heuristic methods that balance class representation through random replication of the minority class and random elimination of majority class examples respectively. There are some limitations of both random under sampling and random over sampling. For instance, under-sampling can discard potentially useful data while over-sampling can increase the likelihood of overfitting [1]. Despite these limitations, random over sampling in general is among the most popular sampling techniques and provides competitive results when compared with most complex methods [1,12].

Several heuristic methods are proposed to overcome these limitations including Tomek links [13], Condensed Nearest Neighbour Rule (CNN) [7], One-sided selection [10] and Neighbourhood Cleaning rule (NCL) [11] are several well-known methods for under-sampling while Synthetic Minority Over-Sampling Technique (SMOTE) is a well-known method for over-sampling technique [5]. The main idea in SMOTE is to generate synthetic examples by operating in the "feature space" rather than the "data space" [5]. The minority class is oversampled by interpolating between several minority class examples that lie together. Depending upon the amount of over-sampling required, neighbours from the k nearest neighbours are randomly chosen. Thus, the overfitting problem is avoided and the decision boundaries for the minority class are spread further into the majority class space [1].

Liu et al. [12] and Chan et al. [3] examine the class imbalance problem by combining classfiers built from multiple under-sampled training sets. In both approaches, several subsets from the majority class with each subset having approximately the same number of samples as the minority class are created. One classifier is trained from each of these subsets and the minority class and then the classifiers are combined. Both these approaches differ in grouping multiple classifiers and in creating subsets from the majority class.

3 Inverse Random under Sampling

In this section, we will discuss the proposed inverse random under sampling (IRUS) method. For convenience, we refer to the minority class as the concept class and the majority class as the anti-concept class. A conventional training of a concept detector using a data set containing representative proportions of samples from the concept and anti concept classes will tend to find a solution that will be biased towards the larger class. In other words, the probability of misclassifying samples from the anti-concept class will be lower than the probability of error for the concept class. However, the actual performance will be determined by the underlying overlap of the two classes and the class prior probabilities. Thus, we need to control the probability of misclassification of samples from the anti-concept class to achieve the required target performance objectives. This may require setting the operating point of the detector so as to achieve false positive rate that is lower than what would be yielded by conventional training. This could be achieved by biasing the decision boundary in favour of the anti concept sample error rates using threshold (off set) manipulation. Alternatively, we could increase the imbalance between the number of samples from the two classes artificially by eliminating some of them. The latter solution is not very sensible, as we would be depleting the class which is naturally underrepresented even further. The former solution would lead to a substantial increase in the false negative rate.

The problem of learning decision functions in situations involving highly imbalanced class sizes is sometimes mitigated by stratified sampling. This aims to create a training set containing a comparable numbers of samples from all the classes. Clearly, in stratified sampling the training set size would be determined by the number of samples in the underrepresented class. This would lead to a drastic subsampling of the anti-concept class with the resultant reduction in the accuracy of the estimated class boundary. This loss of accuracy can be recovered by means of multiple classifier methodology. By drawing randomly multiple subsets from the anti-concept class data set, each adhering to the stratified sampling criteria, we can design several detectors and fuse their opinions. For a typical imbalance of priors of say 100 : 1, the number of the designs would be too low to allow an alternative approach to controlling false positive error rate and one would have to resort to the biasing methods discussed earlier.

Suppose we take the data set manipulation to the extreme and inverse the imbalance between the two classes. Effectively we would have to draw sample sets from the anti-concept class of size proportional to P^2 where P is the prior probability of the concept class. This would lead to very small sample sets for

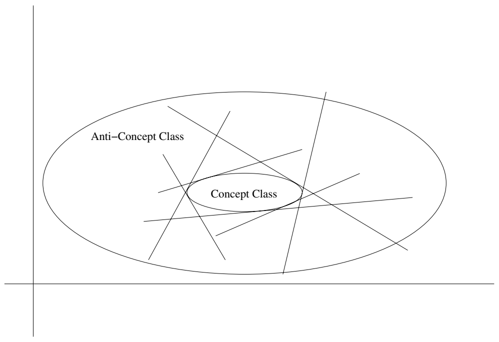


Fig. 1. Schematic diagram showing each boundary partitions the training data set by a hyperplane tangent to the surface of the volume occupied by the concept class

the anti-concept class and therefore, a poor definition of the boundary between the two classes. Nevertheless, the boundary would favour the concept class. Also, as the number of samples from the negative class is very small in relation to the dimensionality of the feature space, the capacity of each boundary to separate the classes fully is high. Moreover, as the number of samples drawn is proportional to P^2, the number of independent sets that can be drawn will be of the order of $\frac{1}{P^2}$. This large number of designs could then be used for controlling the false positive rate using a completely different mechanism. By combining the designed detectors using voting, we can control the threshold on the number of votes needed to accept the concept hypothesis, thus controlling the false positive error rate. This contrasts with the complex task of biasing a decision boundary in high dimensional space.

Interestingly, there is another important benefit of the the IRUS method. As the number of samples forming the negative class is very small, each detector design will be significantly different. This will produce highly diverse detectors which are required for effective classifier fusion. The fused decision rule achieves better class separation than a single boundary, albeit estimated using more samples. This is conveyed schematically in Figure 1. Each boundary partitions the training data set by a hyperplane tangent to the surface of the volume occupied by the concept class. It is the union of these tangent hyperplanes created by fusion, which constitutes a complex boundary to the concept class. Such boundary could not easily be found by a single linear discriminant function. If one resorted to nonlinear functions, the small sample set training would most likely lead to a over fitting and, consequently, to poor generalisation on the test set. Figure 2 provides supporting evidence for the above conjecture. The histogram of discriminant function values (i.e. distance from the decision boundary) generated by one thousand classifiers designed using the inverse imbalance sampling principle for a single negative class test sample (blue bar) shows many of the classifiers scoring positive values which lie on the concept class side of the boundary. This is expected for more than half of the classifiers, as the negative sample will lie beyond the concept class, but nevertheless on the same side as the concept class.

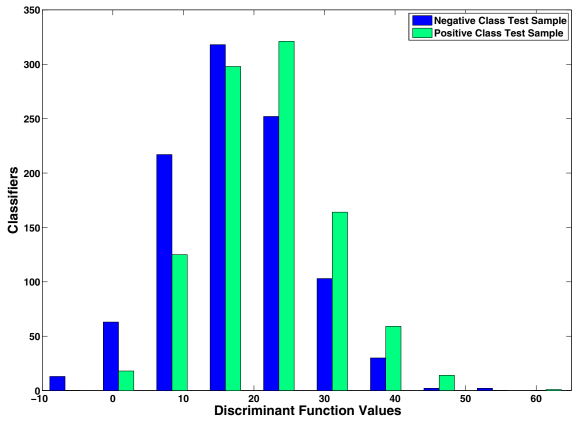


Fig. 2. Histogram of Discriminant Function generated by one thousand classifiers

In contrast, discriminant function values for a single positive class test sample show that most of the classifiers scoring positive values lie on the concept side of the boundary.

In summary, we propose a classifier design approach which is based on an inverse imbalance sampling strategy. This is accomplished by setting the appropriate threshold in the fusion stage combining the outputs of the multiple concept detectors. It allows a very accurate definition of the boundary between the concept class and the negative class.

The pseudo code of IRUS is shown in Algorithm 1. S and $Sets$ are user specified parameters. S controls the number of negative samples drawn at random in each model while $Sets$ determine the number of models or classifiers. For each set Ξ'_a paired with Ξ_c we learn a model h_i. For each model h_i, the probability of unseen instances belonging to concept class D_c is calculated. The probabilities from all models are added. The output is a probability set Ξ_p of the test instances belonging to concept class. Ξ_p is then used to calculate the performance measure discussed in Section 4.3.

4 Experiments

4.1 Experimental Setup

To evaluate the effectiveness of the proposed method, extensive experiments were carried out on 11 public data sets from UCI repository which have different degrees of imbalance [2]. Table 1 describes the data sets used in this study. For each data set, it shows the number of attributes (A), number of samples (N_s), number of majority samples (N_a) and number of minority samples (N_c). As in [1,12], for more than two classes, the class with fewer samples is chosen as the positive class and the remaining as the negative class.

Algorithm 1. PseudoCode for Inverse Random Under Sampling (IRUS)

```
Require: $\Xi_c$: Training set of concept patterns with cardinality $N_c$
  $\Xi_a$: Training set of anti-concept patterns with cardinality $N_a$
  $\Xi_t$: Test set with cardinality $N_t$
  $S$: Number of samples from $\Xi_a$ for each Model
  $Sets$: Number of classifiers, default: $ceil(\frac{N_c}{S})$
Ensure: $\Xi_p$: Probability set of Test instances belonging to concept class
  $\Xi_p \Leftarrow 0$
  for $i = 1$ to $Sets$ do
    $\Xi'_a \Leftarrow$ Randomly pick $S$ samples without replacement from $\Xi_a$
    $T_s \Leftarrow \Xi'_a + \Xi_c$
    Train base classifier $h_i$ using $T_s$ samples
    for $j = 1$ to $N_t$ do
      $D_c \Leftarrow$ Probability distribution of Test Sample $\Xi_{tj}$ belonging to concept class
      from $h_i$
      $\Xi_{pj} \Leftarrow \Xi_{pj} + D_c$
    end for
  end for
```

Table 1. Description of Data sets. Ratio is the size of majority class divided by that of minority class.

Data set	Samples N_s	Attributes A	Concept/Anti-Concept	#min/#maj N_a/N_c	Ratio
Flag	194	28	White/Remainder	17/177	10.42
German	1000	20	Bad/Good	300/700	2.33
Glass	214	9	Ve-win-float-proc/Remainder	17/197	11.59
Haberman	306	3	Die/Survive	81/225	2.78
Mf-Mor	2000	6	10/Remainder	200/1800	9.0
Mf-Zer	2000	47	10/Remainder	200/1800	9.0
Nursery	12960	8	Not-recom/Remainder	328/12632	38.51
Phoneme	5404	5	1/0	1586/3818	2.41
Pima	768	8	1/0	268/500	1.87
Satimage	6435	36	4/Remainder	626/5809	9.28
Vehicle	846	18	Van/Remainder	199/647	3.25

For every data set, we perform a 10-fold stratified cross validation. The whole cross validation is repeated 10 times, and the final values are the averages of these 10 cross validation runs.

4.2 Benchmark Methods

Decision tree (C45) is used as the base classifier for the proposed inverse random under sampling technique (IRUS). The IRUS method is compared with the following class imbalance techniques: Random Under Sampling (RUS), Random Over Sampling (ROS) and SMOTE. The WEKA [14] implementation is used for C45 and SMOTE and the k nearest neighbour parameter is set to 5 in SMOTE.

Further, since pruning and unpruned trees can have different effects on learning from imbalanced data sets, all methods are evaluated using both pruned (25% confidence lavel)/unpruned decision trees. The presented method is also compared with Chan and Stolfo's method [3] (ChSt). The only difference is that the number of majority class examples sampled by ChSt method is equal to the number of minority class examples, while the number of majority class examples sampled in this paper is smaller than the number of minority class examples.

4.3 Performance Measure

The area under the receiver operating characteristic curve (AUC) is most commonly used measure for class imbalance data sets [9,12] and is adopted here. The AUC represents the expected performance as a singular scalar. It integrates performance of the learning method over all possible values of false positive rate. The Mann Witney statistic is used to calculate the AUC and is implemented in WEKA [14].

5 Results and Discussion

Table 2 shows the AUC for various data sets using different methods. It is clear from Table 2 that AUC using unpruned tree is higher than AUC using pruned tree. This is due to the fact that pruning can reduce the minority class coverage in the decision trees [4]. On Nursery and Vehicle data sets, all methods have achieved very high AUC (> 0.95) for both pruned and unpruned decision trees. Overall, our proposed IRUS method has increased performance in 7 out of 11 data sets. There is an increase in the performance in all data sets except phoneme, mf-mor, nursery and satimage. For mf-mor, nursery and satimage, the difference is not significant. However, for phoneme, there is a significant decrease in performance when compared with all other methods. This is explained by the fact that number of positive samples is quite high in this data set (1586 out of 3818) and since only few samples from negative class are used to learn a model, some negative samples are always on the wrong side of the boundary. Overall, the average AUC for IRUS is approximately 10.1%, 4.6%, 2.8%, 2.7%, 0.63% better than J48, RUS, ROS, SMOTE, ChSt respectively when unpruned decision tree is used. It should be noted that the minority class is over-sampled at different values for SMOTE and the highest mean AUC is obtained when minority class is over-sampled at 400%. For IRUS, again after experimenting with different run-time paramters, the paramters used are $S = 15$ and $Sets = 1.5 \times ceil(\frac{N_c}{S})$.

Table 3 shows the results of t-test (significance level 0.05) of AUC. The t-test is shown separately for pruned and unpruned trees in the upper and lower triangles respectively. The table clearly indicates that IRUS achieves significant performance gains when compared with other methods. For unpruned decision tree, the t-test reveals that IRUS performs significantly better in 8 out of 11 data sets when compared with ROS and SMOTE and 5 out of 11 when compared with

Table 2. AUC of the compared methods

Data set	Pruning	J48	RUS	ROS	SMOTE	ChSt	IRUS
Flag	yes	0.5000	0.7354	0.7424	0.6592	0.7891	0.7852
	no	0.7089	0.7581	0.7289	0.6926	0.7921	**0.7949**
German	yes	0.7061	0.6969	0.7058	0.7164	0.7254	0.5365
	no	0.7021	0.6950	0.7047	0.7141	0.7234	**0.7668**
Glass	yes	0.5894	0.7036	0.7635	0.7818	0.7899	0.8148
	no	0.6432	0.7078	0.7656	0.7820	0.8121	**0.8169**
Haberman	yes	0.5851	0.6167	0.6320	0.6693	0.6454	0.6555
	no	0.6182	0.6100	0.6367	0.6726	0.6545	**0.6877**
Mf-Mor	yes	0.500	**0.9294**	0.9234	0.9264	0.9286	0.9275
	no	0.5000	0.9284	0.9227	0.9269	0.9281	0.9262
Mf-Zer	yes	0.5980	0.8660	0.8771	0.8754	0.9006	**0.9072**
	no	0.8667	0.8660	0.8771	0.8752	0.9007	0.9065
Nursery	yes	0.9940	0.9606	0.9975	0.9944	0.9898	0.9850
	no	0.9975	0.9743	**0.9982**	0.9973	0.9965	0.9978
Phoneme	yes	0.9127	0.8931	0.9251	0.9174	0.9146	0.8429
	no	0.9151	0.8960	**0.9254**	0.9195	0.9238	0.8596
Pima	yes	0.7756	0.7572	0.7763	0.7717	0.7671	0.8110
	no	0.7788	0.7626	0.7781	0.7747	0.7689	**0.8167**
Satimage	yes	0.9084	0.9095	0.9214	0.9202	0.9454	0.9289
	no	0.9162	0.9109	0.9213	0.9208	**0.9486**	0.9405
Vehicle	yes	0.9770	0.9649	0.9768	0.9740	0.9810	0.9810
	no	0.9769	0.9679	0.9779	0.9758	**0.9850**	**0.9850**
Average	yes	0.7319	0.8197	0.8405	0.8369	**0.8524**	0.8341
	no	0.7840	0.8252	0.8397	0.8411	0.8581	**0.8635**

Table 3. Summary of *t*-test with significance level at 0.05. The upper triangle shows the results with pruned decision tree and the lower triangle shows the results with unpruned decision trees. Each tabular shows the amount of WIN-TIE-LOSE of a method in a row comparing with the method in a column.

	J48	RUS	ROS	SMOTE	ChSt	IRUS
J48	-	4-2-5	0-3-8	1-2-8	1-2-8	3-0-8
RUS	3-3-5	-	1-3-7	2-0-9	1-1-9	2-1-8
ROS	7-4-0	8-2-1	-	4-5-4	2-3-6	3-0-8
SMOTE	7-4-0	9-1-1	2-7-2	-	3-4-4	3-2-6
ChST	9-1-1	9-2-0	8-2-1	5-4-2	-	4-4-3
IRUS	9-1-1	9-1-1	8-1-2	8-2-1	5-4-2	-

ChSt. IRUS is significantly lower in only 1 data set (phoneme) when compared with RUS and SMOTE while only in 2 data sets when compared with ROS and ChSt. For pruned decision tree, IRUS performs significantly better in 8 and 6 data sets when compared with ROS and SMOTE respectively, although the overall average AUC for IRUS is less than ROS and SMOTE (see Table 2).

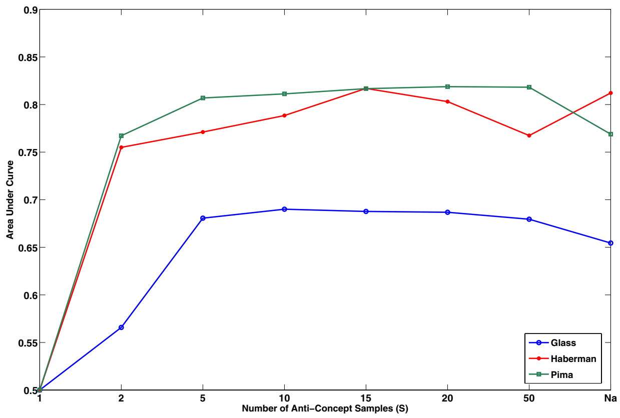


Fig. 3. Run time parameter S vs AUC. N_a = Total number of samples in Concept Class.

Figure 3 shows the different values of run-time parameter S vs AUC in glass, haberman and pima data sets. This parameter effectively controls the number of anti-concept samples drawn at random in each model (classifier). It is observed that IRUS performs best for values in the range $[5 - 20]$. Parameter S also effects the training time. For low value of S, more sets or classifiers (See Algorithm 1) are trained while for high value of S, less classifiers are required. We have also experimented with different values of other run-time parameter $Sets$. This parameter is important to make sure that almost all anti-concept samples are selected during different models. After some experiments, it is observed that the mean AUC is almost identical when $Sets > 1.5 \times ceil(\frac{N_c}{S})$.

6 Conclusion

A novel inverse random under sampling (IRUS) method is proposed in this paper to solve the class imbalance problem. The main idea is to use disproportionate training set sizes, but by inversing the training set cardinalities. By the proposed method of inverse under sampling of the majority class, we can construct a large number of minority class detectors which in the fusion stage has the capacity to realise a complex decision boundary. The distinctiveness of IRUS is assessed experimentally using 11 public UCI data sets. The results indicate significant performance gains when compared with other class imbalance methods.

In this paper, C4.5 is used as a base classifier. It would be interesting to see how other well-known classifiers like NaiveBayes, SVM, KNN, LDA behave when used as a base classifier in our proposed inverse under sampling method.

Acknowledgements. This work was supported by the EU VIDI-Video Project.

References

1. Batista, G., Prati, R.C., Monard, M.C.: A study of the bahavior of several methods for balancing machine learning training data. SIGKDD Explorations 6(20-29) (2004)
2. Blake, C., Keogh, E., Merz, C.J.: UCI repository of machine learning databases
3. Chan, P.K., Stolfo, S.J.: Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In: Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, pp. 164–168 (1998)
4. Chawla, N.V.: C4.5 and imbalanced data sets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In: Proceedings of the International Conference on Machine Learning (ICML 2003) Workshop on Learning from Imbalanced Data Sets II (2003)
5. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Synthetic minority over-sampling technique. Journal of Artificial Inetelligence Review (16), 321–357 (2002)
6. de Souto Marcilio, C.P., Bittencourt, V.G., Jose, A.F.C.: An empirical analysis of under-sampling techniques to balance a protein structural class dataset. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4234, pp. 21–29. Springer, Heidelberg (2006)
7. Hart, P.E.: Condensed nearest neighbor rule. IEEE Transactions on Information Theory 14, 515–516 (1968)
8. Japkowicz, M., Stephen, S.: The class imbalance problem: A systematic study. Intelligent data analysis (6), 429–449 (2002)
9. Kotsiantis, S.B., Pintelas, P.E.: Mixture of expert agents for handling imbalanced data sets. Annals of Mathematics, Computing and Teleinformatics 1(1), 46–55 (2003)
10. Kubat, M., Matwin, S.: Addressing the course of imbalanced training sets: One-sided selection. In: Proceedings of International Conference of Machine Learning, pp. 179–186 (1997)
11. Laurikkala, J.: Artificial Intelligence in Medicine. In: Improving Identification of Difficult Small Classes by Balancing Class Distribution. LNCS. Springer, Heidelberg (2001)
12. Liu, X.Y., Wu, J., Zhou, Z.H.: Exploratory under-sampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics (2009)
13. Tomek, I.: Two modifications of CNN. IEEE Transactions on Systems, Man and Cybernatics (6), 769–772 (1976)
14. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)

Decision Templates Based RBF Network for Tree-Structured Multiple Classifier Fusion

Mohamed Farouk Abdel Hady and Friedhelm Schwenker

Institute of Neural Information Processing
University of Ulm
D-89069 Ulm, Germany
{mohamed.abdel-hady,friedhelm.schwenker}@uni-ulm.de

Abstract. Multiclass pattern recognition problems ($K > 2$) can be decomposed by a tree-structured approach. It constructs an ensemble of K-1 individually trained binary classifiers whose predictions are combined to classify unseen instances. A key factor for an effective ensemble is how to combine its member outputs to give the final decision. Although there are various methods to build the tree structure and to solve the underlying binary problems, there is not much work to develop new combination methods that can best combine these intermediate results. We present here a trainable fusion method that integrates statistical information about the individual outputs (clustered decision templates) into a Radial Basis Function (RBF) network. We compare our model with the decision templates combiner and the existing nontrainable tree ensemble fusion methods: classical decision tree-like approach, product of the unique path and Dempster-Shafer evidence theory based method.

1 Introduction

Many real-world pattern recognition problems contain a large number of classes. Multi-class decomposition schemes, such as the tree-structured approach [1,2,3,4], Error-Correcting Output Codes (*ECOC*) [5], *One-Against-One* and *One-Against-Others* have been considered where an ensemble of classifiers instead of a single classifier is developed. Any multi-class decomposition scheme consists of three stages: (1) decomposition of the multi-class problem into a set of simpler two-class problems, (2) solving these two-class problems and (3) combination of the intermediate solutions to yield the final decision. Ensemble methods can be divided into: *Flat* and *Hierarchical*. Flat architectures are the most popular ones where the members work independently disregarding the hierarchical structure of the classes. *Tree-Structured ensembles* improves the classification performance by taking into account prior knowledge encoded into the class hierarchy.

The main motivations for this paper are: (1) Although there are various methods to build the class hierarchy (first stage) and to solve the underlying binary-class problems (second stage), there is not much work to develop new combination methods that can best combine the intermediate results of the binary classifiers within the hierarchy (third stage). (2) The simple aggregation rules used for flat multiple classifier systems such as *minimum*, *maximum*, *average*, *product* and *majority vote* can not be applied to *hierarchical decision profiles*.

Our main contributions are: (1) A new trainable fusion method for a tree ensemble that integrates statistical information about its individual outputs in the form of decision templates into the training of an RBF network combiner. (2) A new similarity measure based on *multivariate Gaussian function* to match a decision profile with decision templates. (3) The application of the *decision templates* combiner proposed by Kuncheva [6] for hierarchical ensembles.

The remainder of this paper is organized as follows: We give a short description of the tree-structured learning algorithm for multi-class decomposition in Section 2. In Section 3, we discuss briefly the existing approaches for decision fusion for hierarchical ensembles. The proposed combination method is presented in Section 4. Section 5 contains the results of performance evaluation on nine multi-class visual object recognition tasks. Finaly, we conclude the paper in Section 6.

2 Tree-Structured Multi-class Decomposition

Given a training set $L = \{(x_i, y_i)|x_i \in R^D, y_i \in \Omega, i = 1, \ldots, m\}$, the task of the tree-structured approach is to decompose a given K-class problem into a set of K-1 binary-class problems and to train a classifier to solve a binary-class problem at each internal node given a base learning algorithm ($BaseLearn$). At the classification phase, the approach uses a given combination method ($TreeCombiner$) to combine the intermediate results of the internal node classifiers in order to produce the final decision of the ensemble for a given unseen instance x. First, the set of K classes (Ω) is split into two disjoint subsets, known as meta-classes or super-classes. Then the meta-classes are again split recursively until each meta-class contains one of the original classes. The resultant binary tree has K leaf nodes, one for each original class and K-1 internal nodes, each associated with two meta-classes and a binary classifier. (See Algorithm 1).

There are various ways to build the tree structure, e.g. user-defined and class-similarity based approaches. As an example for user-defined approach, in the handwritten digits recognition problem, the user can construct two meta-classes by separating the digits $\{0, 1, 2, 3, 4\}$ in one meta-class and the rest in the other meta-class. If the class hierarchy is based on the relationships between classes, it provides important domain knowledge that leads to improvement in classification accuracy. That is, the class hierarchy should satisfy the well-known cluster assumption: similar classes should belong to the same meta-class while dissimilar classes should belong to different meta-classes. In this study, the Euclidean distance d_{ik} between each pair of class centroids c_i and c_j is used to measure the similarity between classes ω_i and ω_j.

A Top-Down approach is used to construct the binary tree structure. Recursively, for each internal node j, the set of classes Ω_j is divided into two disjoint (dissimilar) subsets Ω_{2j} and Ω_{2j+1} starting from the root node. For example, at

Algorithm 1. Tree Ensemble Learning Algorithm

```
Require: $L$ - set of $m$ labeled training examples
    $\Omega = \{\omega_1, \ldots, \omega_K\}$- set of classes
    $BaseLearn$ - base learning algorithm
    $TreeCombiner$ - hierarchical combination method
    Training Phase
1: $\Omega_1 \leftarrow \Omega$
2: Generate Class Hierarchy as follows:
      1. $C \leftarrow \{(c_k, \omega_k)\}_{k=1}^{K} \leftarrow GetClassCentroids(L)$
      2. $hierarchy \leftarrow BuildNode(\Omega_1, C)$
3: for each internal node $j$ at $hierarchy$ do
4:    Filter the training set $L$ as follows:
      $L_j = \{(x, y) | x \in \Omega_j$ and $y = 0$ if $x \in \Omega_{2j}$ and $y = 1$ if $x \in \Omega_{2j+1}\}$
5:    Train binary classifier, $h_j \leftarrow BaseLearn(L_j)$
6: end for
    Prediction Phase
7: return $TreeCombiner(x, hierarchy)$ for a given instance $x$
```

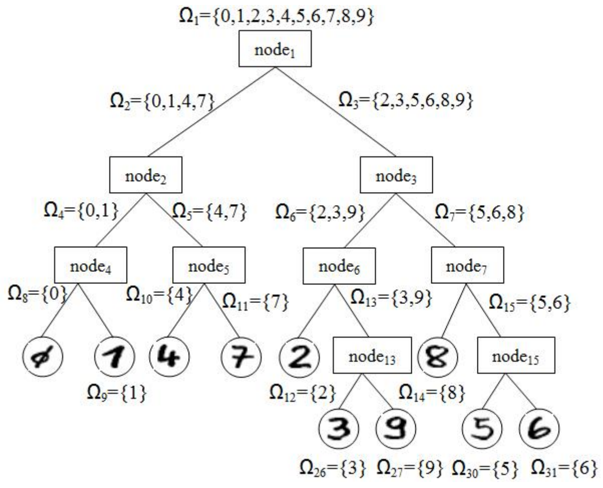


Fig. 1. Class hierarchy for the handwritten digits

the root node with index 1 (see Figure 1), the most distant subsets Ω_2 and Ω_3 of Ω_1 are determined by performing 2-*means clustering* using the centroids of the two most distant classes in Ω_1 as initial prototypes for clusters. Ω_2 and Ω_3 will contain the set of classes lies at the first and second cluster, respectively.

After constructing the tree, a binary classifier h_j is assigned to each internal node j to discriminate between two meta-classes Ω_{2j} and Ω_{2j+1} and it is trained using the filtered training set L_j (See Algorithm 1) .

3 Existing Tree Combination Methods

3.1 Classical Decision Tree-Like (Hard) Combiner

This *Hard Combiner* method assumes that for a given instance x, the internal node classifiers h_j assign a hard class label (hard classifiers), $h_j : R^D \rightarrow \{\Omega_{2j}, \Omega_{2j+1}\}$. Starting from the root node and recursively, the instance is pushed to one of the two child nodes until a leaf node is reached. That is, to assign a class label to a given instance, *only* the classifiers along the unique path from the root node to one of the leaf nodes are considered.

3.2 Product of the Unique Path Combiner

This tree combiner was proposed by Kumar et al. in [1,2] and applied for land-cover classification using remote sensing images. It is based on the assumption that the internal classifier h_j can estimate meta-class membership probabilities (soft classifier). Then, for given instance x, the membership probability for each class k is the product of the posterior probabilities of all the internal classifiers along the unique path from the root node to the leaf node containing class k.

3.3 Dempster-Shafer Evidence Theory Based Combiner

Major advantages of Dempster-Shafer (DS) theory of evidence [7,8] are the ability: (1) to discriminate between ignorance and uncertainty, (2) to represent evidences at different levels of abstraction and (3) to combine evidences produced by different sources. The DS theory starts by assuming a universe of discourse called the *frame of discernment* that consists of a finite set of K mutually exclusive atomic hypotheses $\theta = \{\theta_1, \ldots, \theta_K\}$. Let 2^θ denote the set of all subsets of θ. Then a *mass function* over the frame of discernment θ is a function $m : 2^\theta \rightarrow [0, 1]$ that is called *basic probability assignment* (*bpa*) if it satisfies: $m(\phi) = 0$ and $\sum_{A \subseteq \theta} m(A) = 1$. *Dempster's rule of combination* combines the basic probability assignments produced by n independent sources $m_1, \ldots, m_n$:

$$m(A) = \sum_{\cap A_i = A} \prod_{1 \leq i \leq n} m_i(A_i) \tag{1}$$

DS Theory was used for decision fusion of hierarchical multiclassifier systems by Fay et al in [3,4] where it was applied successfully for visual object recognition.

4 Proposed Tree Combination Method

Now we introduce a new trainable soft fusion method to combine the intermediate results of tree-structured individual classifiers. It is based on the assumption that the combined classifiers have real-valued outputs (soft classifiers). It is inspired from *Stacked Generalization* technique for combining multiple classifiers to improve generalization accuracy introduced by Wolpert [9].

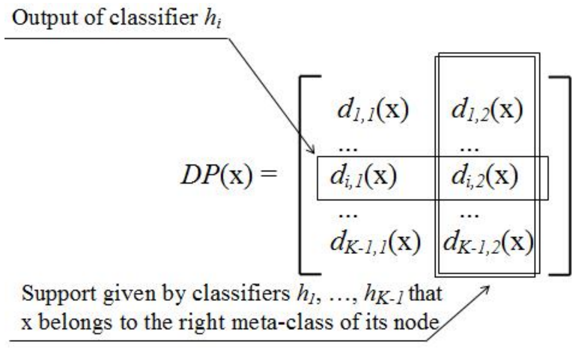

Output of classifier h_i
$DP(x) = \begin{bmatrix} d_{1,1}(x) & d_{1,2}(x) \\ \dots & \dots \\ d_{i,1}(x) & d_{i,2}(x) \\ \dots & \dots \\ d_{K-1,1}(x) & d_{K-1,2}(x) \end{bmatrix}$
Support given by classifiers $h_1, \dots, h_{K-1}$ that x belongs to the right meta-class of its node

Fig. 2. Decision Profile for instance x using tree-structured ensemble classifiers

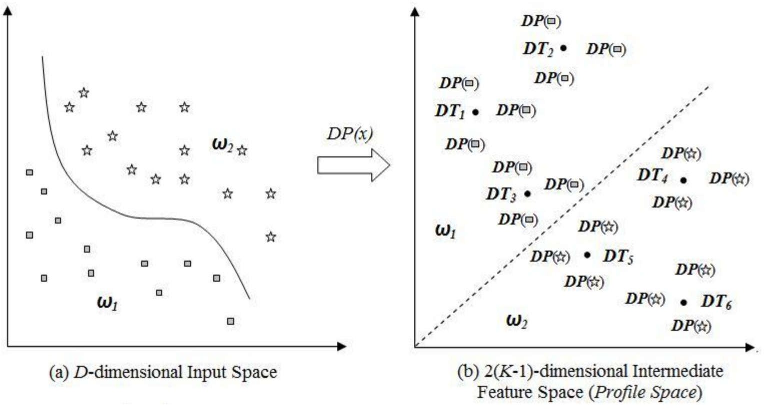

ω_1
ω_2
DP(x)
DT_1
DT_2
DT_3
DT_4
DT_5
DT_6
(a) D-dimensional Input Space
(b) 2(K-1)-dimensional Intermediate Feature Space (Profile Space)

Fig. 3. An illustrative example for data transformation

4.1 Hierarchical Decision Profile

The binary outputs of the K-1 internal node classifiers for each training example x are stored in a decision profile $DP(x)$ as the matrix in Figure 2. Based on the way of using $DP(x)$ to find the overall support for each class k, the fusion methods are divided by Kuncheva [10] into **class-conscious** and **class-indifferent**. **class-conscious** refers to the methods that use the k^{th} column of $DP(x)$ such as average, minimum, maximum and product rules. **class-indifferent** refers to the methods that ignore the context of $DP(x)$ and use all of $DP(x)$ as features in a new feature space, which is called the *intermediate feature space*. From Figure 3, we can observe that the **class-conscious** fusion methods can not be used with the *hierarchical decision profile* because the meta-classes are not the same at different rows. Hence, **class-indifferent** fusion method is required where the final decision of the tree ensemble is made by another classifier that takes the intermediate feature space as input and outputs a class label (see Figure 3).

4.2 Standard Decision Templates Combiner

This trainable combiner was proposed by Kuncheva [6]. At the training phase, a decision template (DT_k) is calculated for each class k as the mean of the decision profiles of the training examples belonging to class k.

$$DT_k = \frac{1}{N_k} \sum_{y_i = k} DP(x_i) \tag{2}$$

At the classification phase, the decision profile for an instance x is matched to the K decision templates using a similarity measure. The class label with the closest decision template will be assigned to x. In [6], Kuncheva discussed 11 different similarity measures and compared them with 14 other techniques. The most popular similarity measures are S_1 measure,

$$\mu_k(x) = S_1(DP(x), DT_k) = \frac{\sum_{i=1}^{K-1} \sum_{j=1}^{2} min(dp(i,j), dt_k(i,j))}{\sum_{i=1}^{K-1} \sum_{j=1}^{2} max(dp(i,j), dt_k(i,j))} \tag{3}$$

and the normalized Euclidean distance,

$$\mu_k(x) = N(DP(x), DT_k) = 1 - \frac{1}{(K-1) \times 2} \sum_{i=1}^{K-1} \sum_{j=1}^{2} (dp(i,j) - dt_k(i,j))^2 \tag{4}$$

This rule is equivalent to applying the *nearest mean classifier* in the profile space.

4.3 RBF Network Combiner Using Clustered Decision Templates

An *RBF network classifier* will be applied in the intermediate feature space instead of the *nearest mean classifier* applied by the above *Decision Templates* combiner. *Multivariate Gaussian function* ϕ_j is used as an RBF at hidden nodes. Since the hidden layer applies a nonlinear transformation to the input data, class separation should be much easier in the hidden unit space (see Figure 3). The output vector y for a given instance x is produced at the final output layer from the weighted summation of the activations of the Gaussian kernels ϕ_j's.

$$y_k = \sum_{j=1}^{K \times c} w_{jk} \phi_j(\|DP(x) - DT_j\|) \;\; where \;\; k = 1, \ldots, K \tag{5}$$

The two-phase learning procedure discussed in [11] is used for training RBF network combiner using the same training set that is used to construct the ensemble members. In the first phase, for each class k, c decision templates are calculated by applying c-means clustering on the decision profiles of all training examples that belong to class k. After this clustering is completed, $K \times c$ clustered decision templates are used as the *RBF* centers. Then the width of the j^{th} *RBF* (σ_j) is set to the distance between the decision template DT_j and the nearest template of different class multiplied by α as in (6) where α should control the degree of overlap between adjacent Gaussian nodes (in our experiments, α=1).

$$\sigma_j = \alpha \min_{i=1,\dots,K\times c} \{\|DT_j - DT_i\|_2 : i \neq j, class(DT_i) \neq class(DT_j)\} \tag{6}$$

Then, the radial basis function ϕ_j is defined as follows,

$$\phi_j(\|DP(x) - DT_j\|) = exp(-\frac{\|DP(x) - DT_j\|_2^2}{2\sigma_j^2}) \tag{7}$$

Then, in the second learning phase the output layer weights W are computed by minimising the MSE at the network output by a matrix pseudoinverse technique using singular value decomposition, $W = H^+T$, where T is the matrix of target outputs of the m training examples where the *1-out-of-K* coding scheme is used and H is the activation matrix,

$$H_{ij} = \phi_j(\|DP(x_i) - DT_j\|)_{j=1,\dots,K\times c}^{i=1,\dots,m} \tag{8}$$

5 Experimental Results

5.1 Methodology

An experimental study is conducted to compare the proposed tree combiner (*RBFN*) with classical decision tree-like approach (*Hard*), product of the unique path combiner (*Product*), Dempster-Shafer evidence theory based combiner (*DS*) and standard *Decision Templates* combiner using S_1 measure (*DT:*S_1) and normalized Euclidean distance (*DT:NM*). The two-phase learning algorithm used to train the *RBFN* tree combiner (see Subsection 4.3) is also used to train the binary RBF network classifier at each node. Except that meta-class specific c-means clustering algorithm (with $c = 10$) is applied independently to the training examples that belong to each meta-class. The nine real-world data sets used in this study are described in Table 1. We intentionally select data sets with variance in number of features (D), number of classes (K) and number of examples (N). All implementation was carried out using the WEKA library [12].

For each data set and tree combiner, 5 runs of 10-fold cross-validation have been performed. The *(Win/Tie/Loss)* record presents three values, the number of data sets for which algorithm A is significantly better, equal, or worse

Table 1. Description of the data sets

ID	Data set	K	D	N	Reference
Fruits1	*Fruits-colorhist-3x3*	7	216	840	Fay [4]
Fruits2	*Fruits-orienthist-sobel-4x4*	7	128	840	Fay [4]
COIL1	*COIL-colorhist-1x1*	20	24	1440	Fay [4]
COIL2	*COIL-orienthist-sobel-2x2*	20	32	1440	Fay [4]
Digits1	*Digits-PCA40*	10	40	2000	Schwenker et al.[11]
Digits2	*Digits-pixels*	10	256	2000	Schwenker et al.[11]
Letters	*Letters*	26	16	2000	UCI Repository [13]
Texture	*Texture*	11	40	1100	UCI Repository [13]
Satimage	*Satimage*	6	36	1286	UCI Repository [13]

than algorithm B with respect to classification accuracy, using *corrected paired t-test* [14] implemented in WEKA at 0.05 significance level. The accuary in our experiments is less than the results on the same data sets reported elsewhere because we use a random subset of the available data to save computation time. Our main concern is the relative accuracy between different combiners.

5.2 Results

Table 2 shows the average test accuracies and standard deviations. For each data set, the highest accuracy achieved is bold faced. The result with bullet($^\bullet$)/open circle($^\circ$) mark indicates that the *RBFN* combiner is significantly better/worse than the respective combiner for the respective data set. We conclude that the *RBFN* combiner significantly outperforms *Hard*, *Product* and *DS* combiners in seven of the nine domains and its behavior is statistically indistinguishable in the remaining two domains. In addition, the *RBFN* combiner is significantly superior to the *DT*:S_1 and *DT:NM* in eight and seven of the nine domains, respectively.

Table 2. Test accuracy using different tree combiners, using 100% of training data

Dataset	*RBFN*(c=3)	*Hard*	*Product*	*DS*	*DT*:S_1	*DT:NM*
Fruits1	**97.05** ± **1.82**	95.95 ± 2.15	96.26 ± 1.94$^\bullet$	96.21 ± 1.99	95.76 ± 2.48	96.52 ± 1.95
Fruits2	**94.90** ± **2.44**	92.21 ± 3.68$^\bullet$	92.79 ± 3.25$^\bullet$	92.86 ± 3.35$^\bullet$	92.64 ± 3.06$^\bullet$	93.67 ± 2.76$^\bullet$
COIL1	**93.89** ± **2.07**	89.08 ± 2.14$^\bullet$	90.81 ± 1.74$^\bullet$	90.54 ± 2.13$^\bullet$	88.60 ± 2.32$^\bullet$	91.33 ± 2.10$^\bullet$
COIL2	**98.75** ± **0.86**	95.94 ± 1.82$^\bullet$	97.72 ± 1.24$^\bullet$	97.21 ± 1.25$^\bullet$	94.15 ± 1.70$^\bullet$	96.54 ± 1.32$^\bullet$
Digits1	**93.58** ± **1.84**	84.19 ± 2.72$^\bullet$	88.88 ± 2.37$^\bullet$	87.68 ± 2.56$^\bullet$	91.82 ± 2.06$^\bullet$	92.13 ± 2.16$^\bullet$
Digits2	**94.46** ± **1.61**	92.11 ± 2.03$^\bullet$	92.61 ± 1.50$^\bullet$	92.90 ± 1.64$^\bullet$	92.38 ± 1.80$^\bullet$	93.24 ± 1.53$^\bullet$
Letters	**80.37** ± **2.74**	68.74 ± 4.34$^\bullet$	73.29 ± 3.93$^\bullet$	72.11 ± 4.07$^\bullet$	71.15 ± 3.22$^\bullet$	73.24 ± 3.12$^\bullet$
Texture	**96.27** ± **1.74**	94.45 ± 1.92$^\bullet$	95.73 ± 1.88	95.05 ± 1.87$^\bullet$	93.65 ± 2.07$^\bullet$	94.45 ± 1.68$^\bullet$
Satimage	**87.92** ± **2.53**	87.49 ± 2.39	87.68 ± 2.61	87.67 ± 2.56	86.17 ± 2.96$^\bullet$	86.59 ± 2.83
ave.	**93.02**	88.91	90.64	90.25	89.59	90.86
(Win/Tie/Loss)		(0/2/7)	(0/2/7)	(0/2/7)	(0/1/8)	(0/2/7)

5.3 Influence of the Training Set Size

One might expect that the performance of *RBFN* combiner would be very poor with small training sets. To study the influence of the training set size, we evaluate the different tree combiners using only 40% of the available training set (see Table 3). The significance is again indicated with bullets and open circles. From the results, we conclude that sample size has no apparent influence on the benefits of *RBFN* combiner because it still works very well with small samples. That is, *RBFN* combiner significantly outperforms *Hard*, *Product*, *DS*, *DT*:S_1 and *DT:NM* combiners in 8, 6, 7, 9, 7 of the nine domains, respectively.

5.4 Influence of the Number of Decision Templates per Class

In all the previous experiments, *RBFN* combiner was trained with 3 clustered decision templates per class (c=3). To study the influence of the number of decision templates per class, we measured test accuracies of the *RBFN* combiner using one, 7, 10, 15 and 20 decision templates per class (see Table 4). The significance

Table 3. Average test accuracy for different tree combiners, using 40% of training data

Dataset	*RBFN*(c=3)	*Hard*	*Product*	*DS*	*DT:S*$_1$	*DT:NM*
Fruits1	**96.10 ± 1.88**	94.74 ± 2.26$^\bullet$	94.95 ± 2.33	95.21 ± 2.31	94.31 ± 2.03$^\bullet$	95.24 ± 2.11
Fruits2	**92.83 ± 3.11**	90.62 ± 3.09$^\bullet$	89.86 ± 3.79$^\bullet$	90.31 ± 3.69$^\bullet$	90.05 ± 3.56$^\bullet$	90.69 ± 3.55$^\bullet$
COIL1	**93.15 ± 2.27**	86.83 ± 2.53$^\bullet$	88.68 ± 2.42$^\bullet$	88.54 ± 2.27$^\bullet$	86.96 ± 2.67$^\bullet$	89.35 ± 2.76$^\bullet$
COIL2	**97.94 ± 1.14**	93.93 ± 2.00$^\bullet$	95.82 ± 1.58$^\bullet$	95.46 ± 1.74$^\bullet$	91.88 ± 2.09$^\bullet$	94.61 ± 1.83$^\bullet$
Digits1	**92.01 ± 1.81**	82.09 ± 2.97$^\bullet$	86.58 ± 2.72$^\bullet$	85.10 ± 2.74$^\bullet$	89.95 ± 1.99$^\bullet$	90.30 ± 1.85$^\bullet$
Digits2	**93.78 ± 1.64**	91.27 ± 2.03$^\bullet$	91.68 ± 2.05$^\bullet$	91.90 ± 1.87$^\bullet$	91.49 ± 1.97$^\bullet$	92.72 ± 1.73$^\bullet$
Letters	**78.21 ± 3.05**	65.06 ± 4.06$^\bullet$	69.70 ± 3.43$^\bullet$	68.66 ± 3.66$^\bullet$	68.66 ± 3.49$^\bullet$	70.66 ± 3.79$^\bullet$
Texture	**95.33 ± 1.89**	93.53 ± 1.94$^\bullet$	94.33 ± 2.18	94.02 ± 2.01$^\bullet$	92.51 ± 2.41$^\bullet$	93.38 ± 2.15$^\bullet$
Satimage	**87.32 ± 3.12**	86.19 ± 2.93	86.58 ± 2.95	86.44 ± 2.98	85.70 ± 3.13$^\bullet$	86.06 ± 3.40
ave.	**91.85**	87.14	88.69	88.40	87.94	89.22
(Win/Tie/Loss)		(0/1/8)	(0/3/6)	(0/2/7)	(0/0/9)	(0/2/7)

Table 4. Average test accuracy for *RBF Network* combiner with different number of clustered decision templates per class (c), using 100% of training data

Dataset	*RBFN*(c=3)	*RBFN*(c=1)	*RBFN*(c=7)	*RBFN*(c=10)	*RBFN*(c=15)	*RBFN*(c=20)
Fruits1	97.05 ± 1.82	96.83 ± 1.87	97.31 ± 1.78	97.38 ± 1.61	**97.52 ± 1.81**	97.36 ± 1.76
Fruits2	94.90 ± 2.44	93.67 ± 2.81$^\bullet$	95.14 ± 2.42	95.24 ± 2.49	95.26 ± 2.58	95.55 ± 2.41
COIL1	93.89 ± 2.07	92.25 ± 2.17$^\bullet$	95.33 ± 1.90°	96.15 ± 1.83°	96.68 ± 1.71°	**96.74 ± 1.61**$^\circ$
COIL2	98.75 ± 0.86	97.12 ± 1.32$^\bullet$	99.57 ± 0.52°	99.76 ± 0.41°	99.88 ± 0.30°	**99.92 ± 0.27**$^\circ$
Digits1	93.58 ± 1.84	92.67 ± 1.98$^\bullet$	94.10 ± 1.71	94.20 ± 1.72	94.13 ± 1.66	**94.26 ± 1.74**
Digits2	94.46 ± 1.61	93.76 ± 1.67$^\bullet$	94.70 ± 1.50	94.78 ± 1.64	94.86 ± 1.63	**94.96 ± 1.51**
Letters	80.37 ± 2.74	75.14 ± 3.42$^\bullet$	83.93 ± 2.65°	85.06 ± 2.93°	86.36 ± 2.42°	**87.27 ± 2.75**$^\circ$
Texture	96.27 ± 1.74	95.13 ± 1.91$^\bullet$	96.93 ± 1.62	97.05 ± 1.50	97.25 ± 1.37	**97.27 ± 1.54**
Satimage	87.92 ± 2.53	87.59 ± 2.69	**88.27 ± 2.35**	88.20 ± 2.35	88.13 ± 2.26	87.92 ± 2.35
ave.	93.02	91.57	93.92	94.20	94.45	**94.58**
(Win/Tie/Loss)		(0/2/7)	(3/6/0)	(3/6/0)	(3/6/0)	(3/6/0)

is again indicated with bullets and open circles. From the results, we can conclude that *RBFN* combiner with $c > 3$ significantly outperforms *RBFN* with c=3 in only three data sets and the improvement is insignificant in the remaining domains. In addition, *RBFN* combiner with c =3 significantly outperforms *RBFN* with c=1 in seven of the nine tasks. Although the *RBFN* combiner with c=1 and both *Decision Templates* combiners are trained with one decision template per class, *RBFN* combiner outperforms both *Decision Templates* combiners (*DT:S*$_1$ and *DT:NM*) due to its trainable output layer and nonlinear behaviour.

6 Conclusions

In this paper, we present a new soft trainable fusion method for tree-structured multiple classifier systems used in problems with a large number of classes. The proposed model integrates statistical information about the individual binary classifier outputs (in the form of *clustered decision templates*) into an RBF network combiner. Multivariate Gaussian function was used as similarity measure to match a *hierarchical decision profile* with decision templates. Not only RBF network was used as combiner but also it was used to construct the ensemble classifiers. We conduct experiments on nine real-world multi-class object recognition tasks including digits, letters, fruits and textures. The experiments have shown that the *RBF Network* tree combiner significantly outperforms the three existing nontrainable tree combiners and the *decision templates* combiner proposed by Kuncheva. We also demonstrate that this combiner is robust to changes in the training set size and the number of decision templates per class.

Acknowledgements

This work was done within the Transregional Collaborative Research Centre SFB/TRR 62 *Companion-Technology for Cognitive Technical Systems* funded by the German Research Foundation (DFG) and DFG grant SCHW623/4-3. The first author was supported by a scholarship of the German Academic Exchange Service (DAAD).

References

1. Kumar, S., Ghosh, J., Crawford, M.: A hierarchical multiclassifier system for hyperspectral data analysis. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 270–279. Springer, Heidelberg (2000)
2. Kumar, S., Ghosh, J., Crawford, M.: Hierarchical fusion of multiple classifiers for hyperspectral data analysis. International Journal of Pattern Analysis and Applications 5(2), 210–220 (2002)
3. Fay, R., Schwenker, F., Thiel, C., Palm, G.: Hierarchical neural networks utilising Dempster-Shafer evidence theory. In: Schwenker, F., Marinai, S. (eds.) ANNPR 2006. LNCS (LNAI), vol. 4087, pp. 198–209. Springer, Heidelberg (2006)
4. Fay, R.: Feature selection and information fusion in hierarchical neural networks for iterative 3D-object recognition. Ph.D thesis, Ulm University (2007)
5. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2(1), 263–286 (1995)
6. Kuncheva, L., Bezdek, J., Duin, R.: Decision templates for multiple classifier fusion: An experimental comparison. Pattern Recognition 34(2), 299–314 (2001)
7. Dempster, A.P.: A generalization of bayesian inference. Journal of the Royal Statistical Society, 205–247 (1968)
8. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)
9. Wolpert, D.H.: Stacked generalization. Neural Networks 5(6), 1289–1301 (1994)
10. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
11. Schwenker, F., Kestler, H., Palm, G.: Three learning phases for radial basis function networks. Neural Networks 14, 439–458 (2001)
12. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (1999)
13. Blake, C., Merz, C.: UCI repository of machine learning databases. University of California (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
14. Nadeau, C., Bengio, Y.: Inference for the generalization error. Machine Learning 62, 239–281 (2003)

Efficient Online Classification Using an Ensemble of Bayesian Linear Logistic Regressors

Narayanan U. Edakunni and Sethu Vijayakumar

School of Informatics, University of Edinburgh
n.u.edakunni@sms.ed.ac.uk, sethu.vijayakumar@ed.ac.uk

Abstract. We present a novel ensemble of logistic linear regressors that combines the robustness of online Bayesian learning with the flexibility of ensembles. The ensemble of classifiers are built on top of a Randomly Varying Coefficient model designed for online regression with the fusion of classifiers done at the level of regression before converting it into a class label using a logistic link function. The locally weighted logistic regressor is compared against the state-of-the-art methods to reveal its excellent generalization performance with low time and space complexities.

1 Introduction

Research in classification has been dominated by kernel based methods like Support Vector Machine (SVM)[9] and more recently by non-parametric methods like Gaussian Process Classification (GP)[6]. Non-parametric methods like GP derives its success by using a covariance function of the input to model the dependency amongst the responses. The response for a test input is then computed as a linear smooth of all the training responses. This in turn leads to a large overhead in the time and space complexities for training and prediction. The need to store away all of the training points in order to produce a prediction for an unseen input makes the kernel machines ill suited for online learning. On the other hand ensemble learning paradigm is ideal for online learning where the learning model has to adjust its complexity in tune with the training data. When new data is observed from a new region of input space an ensemble learner can add a new model to the region of space and adjust its parameters to model that particular region of space. To implement such an ensemble we need classifiers that have varying responsibilities in different regions of space and are able to learn independent of each other. Conventional ensemble learners combine the classifier predictions either by a majority voting or by linear combination of the votes. This would not work when the ensembles have different levels of confidence over the input space. The combined prediction would be more robust if the predictions are weighted by confidence of individual classifiers before combining them. In this paper we use a linear logistic regression as the base classifier with Bayesian learning for the regression. The combination of the predictions is done at the regression level wherein each learner is endowed with a predictive distribution and the variances of the distribution are used to weigh the predictions of the base learners. The real valued output of the regression is then converted to a class label using a logistic link function. In this paper we combine the robustness and efficiency of a Bayesian learning with the flexibility of ensemble learning to produce an online classifier that is able to adapt its complexity in tune with the observed data. We reuse a Randomly Varying Coefficient model [3] designed for regression to build our ensemble classifier system.

2 Randomly Varying Coefficient Model

The Randomly Varying Coefficient model approximates a multivariate non-linear function using a set of local models. Each of the local models has a probabilistic formulation with a parametric model for the *linear fit* and the *extent* of linearity at a particular location in the input space. The strength of RVC arises from the fact that the local models are trained independent of each other unlike [5]. Apart from minimizing the interference between two models, this also allows models to be dynamically allocated and deallocated without the need for relearning. Furthermore, the probabilistic formulation of RVC provides an estimate of the uncertainty in its prediction and allows Bayesian inference rules to be applied in order to learn the parameters.The end result of an RVC is a set of linear regression models distributed in the input space that have different confidence in their predictions in different regions of the input space.

3 Local Logistic Regression

In this section, we introduce the probabilistic model for a locally weighted logistic regression based on the probabilistic formulation of RVC called the *logistic RVC (lRVC)*. Here, the probability of a binary class variable z_i is modeled as the output of a logistic link function over a continuous latent variable y_i as :

$$p(z_i = 1|y_i) = 1/(1 + exp(-y_i)), \quad i = 1 \dots N \tag{1}$$

where i is the index over the training data. In turn, the latent variable y_i is modeled as the response of a locally linear regression that follows the RVC formulation. For a locally linear region centered around $\mathbf{x}_c$ a conditional model for the continuous latent variable y_i can be written as:

$$y_i = \boldsymbol{\beta}_i^T \mathbf{x}_i + \epsilon \tag{2}$$

where $\mathbf{x}_i \equiv [(\mathbf{x}'_i - \mathbf{x}_c)^T, 1]^T$ represents the center subtracted, bias augmented d dimensional input vector, $\boldsymbol{\beta}_i \equiv [\beta_i^{(1)} \dots \beta_i^{(d+1)}]^T$ represents the corresponding regression coefficient and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian mean zero noise with a variance σ^2. Crucially the latent regression coefficient $\boldsymbol{\beta}_i$ has a Gaussian distribution :

$$\boldsymbol{\beta}_i \sim \mathcal{N}(\hat{\boldsymbol{\beta}}, \mathbf{C}_i) \tag{3}$$

where the magnitude of the covariance $\mathbf{C}_i$ is made proportional to the distance of $\mathbf{x}'_i$ from the center by modeling it as a diagonal covariance with the elements being a quadratic function of the input :

$$\mathbf{C}_i(j, j) = \mathbf{x}_i^T \mathbf{x}_i / h_j^2, \quad j = 1 \dots d \tag{4}$$

This has the effect that for points that lie near the center, the latent regression coefficients $\boldsymbol{\beta}_i$ have similar values, with the distribution of $\boldsymbol{\beta}_i$ being peaked around $\hat{\boldsymbol{\beta}}$. This in turn results in a linear region around the center.

Proceeding along the same lines as Chapter 3 of [6] we now assume a noise-free latent variable y_i by setting σ^2 to zero. Setting σ^2 to zero yields the following model for y_i :

$$y_i \sim \mathcal{N}(\boldsymbol{\beta}_i^T \mathbf{x}_i, 0) \tag{5}$$

or equivalently marginalizing y_i :

$$p(z_i = 1|\boldsymbol{\beta}_i) = 1/(1 + exp(-\boldsymbol{\beta}_i^T \mathbf{x}_i)) \tag{6}$$

which corresponds to the classical formulation of a linear logistic regression with regression coefficients $\boldsymbol{\beta}_i$.

We preserve the same probabilistic model as the original RVC for the rest of parameters. This includes a Gamma distribution as a *regularizer* prior over the bandwidth parameters :

$$h_j^2 \sim Gamma(a_j, b_j) \tag{7}$$

and a noninformative Normal prior $\mathcal{N}(\boldsymbol{\mu}, \mathbf{S})$ for the parameter $\hat{\boldsymbol{\beta}}$.

We can now infer the posterior distribution over the parameters of the model using a Variational Bayesian EM similar to [3]. In this procedure, the posterior distribution over the parameters are assumed to be independent and these distributions are iteratively determined one at a time by fixing all other distributions.

For a logistic regression, an additional complication arises in the computation of the posterior distribution over the hidden variables $\boldsymbol{\beta}_i$. The likelihood term $P(z_i|\boldsymbol{\beta}_i)$ is given by a logistic link function whereas the prior over $\boldsymbol{\beta}_i$ is Gaussian and is not conjugate to the likelihood term. We solve this issue by using a Laplacian approximation to approximate the posterior over $\boldsymbol{\beta}_i$ by a Gaussian distribution. The log posterior over the hidden variable $\boldsymbol{\beta}_i$ is given by :

$$\begin{aligned}\mathcal{M} = \ln Q(\boldsymbol{\beta}_i|\mathbf{z}) &= \ln P(z_i|\boldsymbol{\beta}_i) + \left\langle \ln P(\boldsymbol{\beta}_i|\hat{\boldsymbol{\beta}}, \mathbf{C}_i) \right\rangle_{Q(\hat{\boldsymbol{\beta}}),Q(\mathbf{h})} \\ &- \ln \int exp\left(\ln P(z_i|\boldsymbol{\beta}_i) + \left\langle \ln P(\boldsymbol{\beta}_i|\hat{\boldsymbol{\beta}}, \mathbf{C}_i) \right\rangle_{Q(\hat{\boldsymbol{\beta}}),Q(\mathbf{h})} \right)\end{aligned} \tag{8}$$

Laplace approximation of the posterior corresponds to

$$Q(\boldsymbol{\beta}_i|\mathbf{z}) \sim \mathcal{N}(\boldsymbol{\nu}_i, \mathbf{G}_i)$$

where $\boldsymbol{\nu}_i = argmax_{\boldsymbol{\beta}_i} \mathcal{M}$ and $\mathbf{G}_i^{-1} = -\nabla\nabla\mathcal{M}|_{\boldsymbol{\beta}_i = \boldsymbol{\nu}_i}$ is the Hessian of the negative log posterior. The posterior mode $\boldsymbol{\nu}_i$ can be obtained by setting the gradient of $\mathcal{M}$ to zero. However, this procedure does not yield a closed form solution for the posterior mode $\boldsymbol{\nu}_i$. We then have to resort to Newton's update to find the mode iteratively as shown :

$$\boldsymbol{\nu}_i = \boldsymbol{\nu}_i^{old} - (\nabla\nabla\mathcal{M})^{-1}\nabla\mathcal{M} \tag{9}$$

Substituting the forms of $P(\boldsymbol{\beta}_i|\hat{\boldsymbol{\beta}}, \mathbf{C}_i)$ and $P(z_i|\boldsymbol{\beta}_i)$ from eqs. (3) and (6) into eq. (8) and differentiating it with respect to $\boldsymbol{\beta}_i$ we get :

$$\nabla\mathcal{M}\mid_{\boldsymbol{\beta}_i = \boldsymbol{\nu}_i^{old}} = \mathbf{x}_i(z_i - \pi_i) - \langle \mathbf{C}_i \rangle^{-1} (\boldsymbol{\nu}_i^{old} - \tilde{\boldsymbol{\mu}}) \tag{10}$$

$$\nabla\nabla\mathcal{M}\mid_{\boldsymbol{\beta}_i = \boldsymbol{\nu}_i^{old}} = -\mathbf{x}_i \mathbf{x}_i^T \pi_i(1 - \pi_i) - \langle \mathbf{C}_i \rangle^{-1} \tag{11}$$

Algorithm 1. Training a local model

```
1: Initialize hyperparameters: $\Theta \equiv \{\boldsymbol{\mu}_0, \mathbf{S}, \mathbf{a}, \mathbf{b}\}$.
2: Input: Batch training data $\mathbf{X}$, $\mathbf{z}$
3: repeat
4:    Initialize $\boldsymbol{\nu}_i^{old} = \tilde{\boldsymbol{\mu}}$, $\pi_i = 1/(1 + exp(-\mathbf{x}_i^T \boldsymbol{\nu}_i^{old}))$ and $w_i = \frac{1}{\pi_i(1-\pi_i)}$.
5:    Estimate posterior hyperparameters $\tilde{\Theta}$ using $\Theta$ and eqs. (13) - (16).
6:    Estimate values of the hyperparameters a and b of the regularizer prior using
      eq. (17).
7: until convergence of $\tilde{\Theta}$
```

where $\pi_i = 1/(1 + exp(-\mathbf{x}_i^T \boldsymbol{\nu}_i^{old}))$ and $\langle \mathbf{C}_i \rangle = diag(\mathbf{x}_i^T \mathbf{x}_i / \langle h_j^2 \rangle_{Q(h_j^2)})$. It can be found from eq. (11) that $\mathbf{G}_i^{-1} = -\nabla\nabla\mathcal{M} = \mathbf{x}_i \mathbf{x}_i^T \pi_i(1-\pi_i) + \langle \mathbf{C}_i \rangle^{-1}$ and the estimate for $\boldsymbol{\nu}_i$ can be obtained by substituting eqs. (10) and (11) in eq. (9) yielding :

$$\boldsymbol{\nu}_i = \boldsymbol{\nu}_i^{old} + \mathbf{G}_i(\mathbf{x}_i(z_i - \pi_i) - \langle \mathbf{C}_i \rangle^{-1} (\boldsymbol{\nu}_i^{old} - \tilde{\boldsymbol{\mu}})) \tag{12}$$

which can be simplified using Sherman-Morrison Woodbury theorem to yield :

$$\mathbf{G}_i = \langle \mathbf{C}_i \rangle - \frac{\langle \mathbf{C}_i \rangle \mathbf{x}_i \mathbf{x}_i^T \langle \mathbf{C}_i \rangle}{w_i + \mathbf{x}_i^T \langle \mathbf{C}_i \rangle \mathbf{x}_i} \tag{13}$$

$$\boldsymbol{\nu}_i = \frac{\langle \mathbf{C}_i \rangle \mathbf{x}_i}{(w_i + \mathbf{x}_i^T \langle \mathbf{C}_i \rangle \mathbf{x}_i)}((z_i - \pi_i)w_i + \mathbf{x}_i^T \boldsymbol{\nu}_i^{old} - \mathbf{x}_i^T \tilde{\boldsymbol{\mu}}) + \tilde{\boldsymbol{\mu}} \tag{14}$$

where $w_i = \frac{1}{\pi_i(1-\pi_i)}$.

Posterior over $\hat{\boldsymbol{\beta}}$ based on its likelihood and prior can be derived similar to [3] :

$$Q(\hat{\boldsymbol{\beta}}|\mathbf{z}) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\mathbf{S}})$$

where

$$\tilde{\mathbf{S}} = (\sum_i \langle \mathbf{C}_i \rangle^{-1} + \mathbf{S}^{-1})^{-1},\ \tilde{\boldsymbol{\mu}} = \tilde{\mathbf{S}}(\sum_i \langle \mathbf{C}_i \rangle^{-1} \boldsymbol{\nu}_i + \mathbf{S}^{-1}\boldsymbol{\mu}) \tag{15}$$

Similarly, the posterior over h_j is given by :

$$Q(h_j^2|\mathbf{z}) \sim Gamma(\tilde{a}_j, \tilde{b}_j)$$

where

$$\tilde{a}_j = a_j + N/2,\ \tilde{b}_j = b_j + \sum_i \left[(\boldsymbol{\nu}_{i,j} - \tilde{\boldsymbol{\mu}}_{i,j})^2 + \mathbf{G}_{i,jj} + \tilde{\mathbf{S}}_{jj} \right] / 2\mathbf{x}_i^T \mathbf{x}_i \tag{16}$$

and the optimal values for a_j and b_j are obtained by an update rule given by -

$$a_j = \tilde{a}_j, \quad b_j = \tilde{b}_j \tag{17}$$

Hence, posterior parameters are inferred by using a partial Newton step to infer the posterior of $\boldsymbol{\beta}_i$ followed by EM updates as shown in Algorithm 1.

We observe from the training updates that the base classifier is extremely efficient with a complexity of $O(dMN)$ for training, where d is the number of dimensions of the input space, M the number of local models and N the number of training instances.

4 Prediction

Using the learning procedure discussed in the previous section we obtain independently trained local models of the logistic regression. Each of the local models represent a separate classifier with a linear decision boundary. To obtain an aggregate prediction for a particular query input we need to combine these classifiers.

Ensemble learning has been a field of research which has seen considerable amount of research into the ways of combining classifiers [4]. In this paper, we use the same technique as RVC - combining the linear regressors to produce a non-linear regression model followed a logistic transform to obtain a classifier.

Given the ensemble of trained local experts, in order to predict the response y_q for a new query point $\mathbf{x}_q$, we take the normalized product of the *predictive distribution* of each local expert. This is similar to the predictive routine in Bayesian Committee Machines [8]. The predictive distribution of each local expert is given by:

$$y_{q,k} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}^T \mathbf{x}_{q,k}, {\mathbf{x}_{q,k}}^T (\tilde{\mathbf{S}}_k + \mathbf{C}_{k\mathbf{h}_{mode}})\mathbf{x}_{q,k})$$

where $\mathbf{x}_{q,k}$ refers to the query point with the k-th center subtracted and augmented with bias. Blending the prediction of different experts by taking their product and normalizing it results in a Normal distribution given by:

$$y_q \sim \mathcal{N}(\mu, \zeta^2) \quad \text{where} \quad \mu = \frac{\sum_k \alpha_k \tilde{\boldsymbol{\mu}}_k^T \mathbf{x}_{q,k}}{\sum_k \alpha_k}, \quad \zeta^2 = \frac{1}{\sum_k \alpha_k}. \tag{18}$$

Here, μ is a sum of the means of each individual expert weighted by the confidence expressed by each expert in its own prediction α_k, ζ^2 is the variance and α_k is the precision of each expert:

$$\alpha_k = 1/(\mathbf{x}_{q,k}^T (\tilde{\mathbf{S}}_k + \mathbf{C}_k)\mathbf{x}_{q,k}) \tag{19}$$

The predictive probability for the logistic regression can be obtained by combining the predictive probability of the latent variable y_q with the link function and marginalizing the latent variable to yield :

$$P(z_q = 1|\mathbf{z}) = \int P(z_q = 1|y_q) P(y_q|\mathbf{z}) dy_q \tag{20}$$

where $P(z_q = 1|y_q)$ is a logistic function and $P(y_q|\mathbf{z})$ is the predictive distribution given by eq. (18). The integral given in eq. (20) cannot be evaluated analytically and we must rely on numerical methods or sampling to evaluate the integral. In the context of binary classification if we threshold the predictive probability at $\frac{1}{2}$ in order to discriminate between classes, a maximum aposteriori(MAP) prediction would be the same as an averaged prediction as shown in [1] and explained in [6]. Therefore we use MAP predictive estimate for classification. To obtain the MAP prediction we evaluate the integral in eq. (20) by approximating $P(y_q|\mathbf{z})$ by a delta function at its mode. The prediction routine is listed in Algorithm 2.

Algorithm 2. Global prediction using local models

```
1: Input: Query point $\mathbf{x}_q$
2: Initialize: $sum_\alpha = 0$, $y_q = 0$
3: for k = 1 to #local models do
4:     $\mathbf{x}_{q,k} = \mathbf{x}_q - \mathbf{x}_{c,k}$
5:     Calculate $\alpha_k$ using eq. (19)
6:     $y_q = y_q + \alpha_k \tilde{\boldsymbol{\mu}}_k^T \mathbf{x}_{q,k}$
7:     $sum_\alpha = sum_\alpha + \alpha_k$
8: end for
9: $y_q = y_q / sum_\alpha$
10: Output : $P(z_q = 1) = 1/(1 + exp(-y_q))$
```

5 Online Classification

We can use the same technique as in the original RVC [3] to convert the batch updates derived earlier into online updates. For this we make use of the fact that in a Bayesian inference posterior is given by :

$$posterior_N = \prod_i^N (likelihood_i) \times prior_0$$

where i is an index over the data points. The same can be expressed as a set of online updates :

$$posterior_i \propto likelihood_i \times prior_i \quad ; \quad prior_{i+1} = posterior_i$$

This set of updates implies that at every step of the online update the prior computed over the data seen so far is combined with the likelihood of the current data point to yield the posterior. This new posterior distribution of the parameter is then used as the prior during the next update. Based on this, we can derive the online updates for the logistic RVC that correspond to the batch results derived earlier :

$$\tilde{\mathbf{S}}_i = (\langle \mathbf{C}_i \rangle^{-1} + \mathbf{S}_i^{-1})^{-1} \tag{21}$$

$$\tilde{\boldsymbol{\mu}}_i = \tilde{\mathbf{S}}_i (\langle \mathbf{C}_i \rangle^{-1} \boldsymbol{\nu}_i + \mathbf{S}_i^{-1} \boldsymbol{\mu}_i) \tag{22}$$

$$\tilde{a}_{i,j} = a_{i,j} + 1/2 \tag{23}$$

$$\tilde{b}_{i,j} = b_{i,j} + \left[(\boldsymbol{\nu}_{i,j} - \tilde{\boldsymbol{\mu}}_{i,j})^2 + \mathbf{G}_{i,jj} + \tilde{\mathbf{S}}_{i,jj} \right] / (2\mathbf{x}_i^T \mathbf{x}_i) \tag{24}$$

where $\boldsymbol{\nu}_i$ and $\mathbf{G}_i$ are given by eq. (13). We repeat the above updates for a single data point $\{\mathbf{x}_i, y_i\}$ till the posteriors $\tilde{\Theta} \equiv \{\tilde{\mathbf{S}}, \tilde{\boldsymbol{\mu}}, \tilde{a}, \tilde{b}\}$ converge. For the $(i+1)$-th point, we then use posterior $\tilde{\Theta}$ of i-th step as the prior $\Theta \equiv \{\mathbf{S}, \boldsymbol{\mu}, \mathbf{a}, \mathbf{b}\}$.

When learning from data in an online fashion, we need to dynamically adapt the model complexity of the learning to reflect the complexity of the data being modeled. This can be accomplished by adding and deleting local models depending on whether we need to increase the model complexity or decrease it. We employ a heuristic similar to [3] for addition and deletion of models. Here a new local model is added when the predictive probability of a class falls below a certain threshold. A local model is pruned when there is sufficient overlap between the local regions of two local models.

6 Evaluation

Before we proceed to detailed evaluation experiments, we need to specify the evaluation measures that would be used to compare different classifiers. We compare classifiers based on two different measures - *misclassification error* and *target information*. The former is the often used loss function that measures the mean number of misclassifications produced by a classifier on a test set. The *target information* criteria refers to a loss function that takes into account the confidence expressed by the classifier about its prediction. The loss function is given by :

$$\mathcal{I} = \frac{1}{N}\left[\sum_{z_i=1} log_2(P(z_i = 1|x_i^q)) + \sum_{z_i=0} log_2(1 - P(z_i = 1|x_i^q))\right] + 1 \tag{25}$$

and it measures in bits, the information conveyed by the classifier about the test target. For a baseline classifier that assigns classes at random $\mathcal{I} \to 0$ and for a more confident discrimination of classes $\mathcal{I} \to 1$. It must be noted that this measure has a strong penalty for confident misclassification and can lead to $\mathcal{I} < 0$.

6.1 Comparison of Generalization Performance and Efficiency of Learning

In the first evaluation, we compare the generalization performance of lRVC against a Gaussian Process classifier with squared exponential covariance function and a baseline probabilistic linear logistic regressor. The lRVC used in the evaluation used around 20 local models initialised at the cluster centers in the input space. The Gaussian process uses a square exponential kernel and a logistic link function. Hyperparameters of the GP are learnt using a gradient descent. The two classifiers are compared on different benchmark datasets listed in Table 1. The Breast cancer, Heart (Cleveland) and the Ionosphere dataset were obtained from the UCI repository, Pima and synthetic datasets are the same as the ones used in [7][1]. The USPS dataset corresponds to the digit discrimination task listed in [6]. The Catalysis and Gatineau datasets were obtained from the predictive uncertainty challenge [2] were the validation set has been used as test set. The evaluations on the datasets obtained from UCI was carried out on 10 train-test splits of the data and the mean and standard deviations are reported here. For all other datasets a single train-test trial was carried out using the train and test files provided. This makes it possible to compare other classifiers that have previously used the latter datasets. For the Gatineau dataset GP was trained using a subset of 1000 training points due to practical considerations of time and space complexity. The evaluation statistics are listed in Table 2. Also shown in the table is the results for a LIBSVM[2] (with an RBF kernel) evaluation over the same datasets with the parameters being chosen using a 5 fold

[1] The datasets can be obtained from http://www.stats.ox.ac.uk/pub/PRNN/

[2] http://predict.kyb.tuebingen.mpg.de/pages/home.php

Table 1. Statistics of the benchmark datasets

Dataset	#train pts.	#test pts	#dim
Breast cancer	142	427	30
Heart(Cleveland)	149	148	13
Ionosphere	175	176	33
Pima	200	332	7
Synthetic	250	1000	2
USPS(3-5)	767	773	256
Catalysis	873	300	617
Gatineau	3000	2176	1092

Table 2. Performance comparison between lRVC and GP in terms of the misclassification error rate and the target information (measured in bits) conveyed by the classifier. The values in parenthesis indicate the standard deviation.

	lRVC		GP		Linear		SVM
	Error	Information	Error	Information	Error	Information	Error
Breast	0.028(0.007)	0.807(0.010)	0.026(0.009)	0.805(0.045)	0.042(0.014)	0.797(0.050)	0.028(0.009)
Heart	0.166(0.017)	0.432(0.049)	0.169(0.017)	0.423(0.039)	0.173(0.024)	0.388(0.113)	0.173(0.022)
Ionosphere	0.152(0.027)	0.338(0.170)	0.123(0.025)	0.535(0.054)	0.163(0.027)	-2.288(0.963)	0.078(0.025)
Pima	0.202	0.361	0.222	0.276	0.198	0.364	0.198
Synthetic	0.100	0.649	0.093	0.658	0.114	0.611	0.100
USPS(3-5)	0.045	0.798	0.025	0.794	0.040	0.476	0.023
Catalysis	0.303	0.143	0.303	0.150	0.343	-3.516	0.323
Gatineau	0.090	0.570	0.090	0.588	0.154	-0.484	0.090

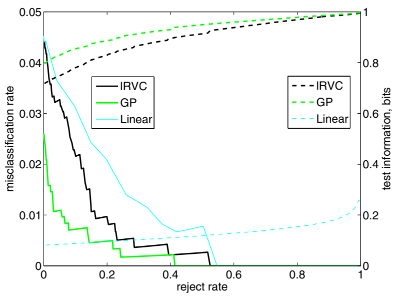
cross validation. The comparison for SVM is restricted to the misclassification error since SVM does not provide a predictive probability. One can see from the results that lRVC is able to match the performance of GP for all the datasets and outperforms the baseline linear classifier especially when the target information is used for the comparison. It must be noted that while lRVC used only a small number of local models for prediction, GP used all of the training set for training and prediction. To emphasize this difference Table 3 shows the time taken by lRVC and GP for training and prediction on a dataset consisting of the USPS digit 3 classified against the rest of the digits. lRVC can be seen to achieve a good generalization performance with a low overhead.

6.2 Use of Predictive Confidence Bounds for Rejection

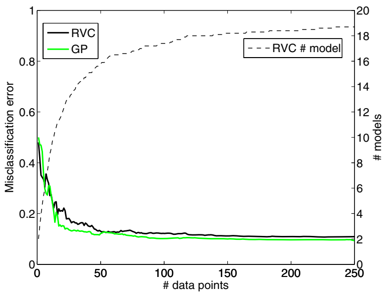
In the next evaluation, we evaluate the confidence bounds learnt by lRVC by plotting the relation between the reject rate, the misclassification error and the target information. In this experiment the first and second moments for the predictive probability were computed by using sampling to evaluate the integral in eq. (20). The test samples which had a variance above a threshold were rejected and the misclassification error was evaluated for the rest of the test data. The dataset used for this purpose was the USPS data. The misclassification error typically decreases as test samples are rejected and an ideal classifier would have a larger reduction in the misclassification error with respect to the rejection rate.

Table 3. Comparison between the time taken for training and prediction using a Matlab implementation of lRVC and GP on the USPS dataset

Method		0	1	2	4	5	6	7	8	9
	# training pts.	1173	1028	881	815	767	826	796	783	828
	# testing pts.	1204	1065	872	861	773	832	820	749	817
GP	Error	0.011	0.002	0.018	0.005	0.026	0	0.002	0.025	0.006
	Information	0.894	0.903	0.863	0.869	0.794	0.886	0.868	0.855	0.849
	Train time(sec)	1915.9	1475.8	1020.8	877.0	911.4	963.7	919.9	845.8	993.0
	Test time(sec)	47.8	37.6	25.0	22.1	20.2	23.8	23.4	20.5	24.4
lRVC	Error	0.009	0.005	0.022	0.005	0.045	0.003	0.009	0.032	0.012
	Information	0.944	0.960	0.886	0.966	0.798	0.972	0.955	0.883	0.946
	Train time(sec)	582.59	509.67	440.44	402.23	382.08	400.75	370.25	363.13	383.42
	Test time(sec)	0.87	0.74	0.61	0.61	0.55	0.55	0.54	0.49	0.55



(a)



(b)

Fig. 1. (a) Comparison of error-reject curve for lRVC and GP (b) Online learning dynamics of lRVC compared with GP. The plots are the average performance over 10 trials of different orders of data presentations.

The error and the target information versus the rejection rate for lRVC, GP and the Bayesian linear logistic regressor were evaluated and plotted in fig. (1(a)). It can be seen that lRVC's performance exceeds that of the linear classifier by a large margin and is not significantly different from GP.

6.3 Dynamics of Online Learning

In the last evaluation, we use the online updates derived in Section 5 to learn a classifier on the synthetic dataset. The data points are presented to the online learner one at a time and the misclassification error is evaluated over the test data after each training update. The dynamics of the learning process is shown in fig. (1(b)). The learning dynamics is compared with the generalization performance of GP which uses increasing number of training data points and the corresponding test error at each stage is displayed. It can be seen from the figure that online version of lRVC exhibits fast convergence and matches the performance of GP asymptotically.

7 Discussion

Local logistic regression is a very competitive method as can be seen from the results of the evaluation. The result makes it more significant when we take it into account that the logistic regression is able to achieve such a good performance using a small number of local models. Moreover the time and space efficiency is linear in terms of the data points and the dimension. In contrast, kernel classification paradigms like GP and SVM have a much higher overhead in training and testing.

The logistic regression formulation in this paper is restricted to binary classification. It can be easily extended to a multi-class classification using a softmax link function instead of a logistic link function. The treatment of the learning remains the same in that case too.

In conclusion, the contribution of this paper has been a probabilistic formulation of a local linear logistic regressor that combines the modeling guarantees of a Bayesian method with the efficiency of an ensemble learner thus making it ideal candidate for online real-time learning.

References

1. Bishop, C.M.: Neural Networks for Pattern Recognition. Claredon Press (1995)
2. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
3. Edakunni, N.U., Schaal, S., Vijayakumar, S.: Kernel carpentry for online regression using randomly varying coefficient model. In: International Joint Conference on Artificial Intelligence, pp. 762–767 (2007)
4. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
5. Nikunj, C.: Oza and Stuart Russell. Online bagging and boosting. In: Artificial Intelligence and Statistics, pp. 105–112 (2001)
6. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
7. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge (1996)
8. Tresp, V.: A Bayesian committee machine. Neural Computation 12(11), 2719–2741 (2000)
9. Vapnik, V.N.: Statistical learning theory. Wiley, New York (1998)

Regularized Linear Models in Stacked Generalization

Sam Reid and Greg Grudic

University of Colorado at Boulder, Boulder CO 80309-0430, USA

Abstract. Stacked generalization is a flexible method for multiple classifier combination; however, it tends to overfit unless the combiner function is sufficiently smooth. Previous studies attempt to avoid overfitting by using a linear function at the combiner level. This paper demonstrates experimentally that even with a linear combination function, regularization is necessary to reduce overfitting and increase predictive accuracy. The standard linear least squares regression can be regularized with an L2 penalty (Ridge regression), an L1 penalty (lasso regression) or a combination of the two (elastic net regression). In multi-class classification, sparse linear models select and combine individual predicted probabilities instead of using complete probability distributions, allowing base classifiers to specialize in subproblems corresponding to different classes.

1 Introduction

Multiple classifier systems combine the predictions of many classifiers to produce the ensemble prediction [1,2,3]. Simple techniques such as voting or averaging can improve predictive accuracy by combining diverse classifiers [4]. More sophisticated ensemble techniques, such as ensemble selection, train a combination function in order to account for the strengths and weaknesses of the base classifiers and to produce a more accurate ensemble model [5].

Stacked generalization is a flexible method for multiple classifier systems in which the outputs of the base-level classifiers are viewed as data points in a new feature space, and are used to train a combiner function [6][1]. Ting and Witten [7] applied stacked generalization to classification problems, and found that a multiple response linear combiner outperformed several nonlinear combiners on their problem domains and selection of base classifiers. They also showed that in classification problems, it is more effective to combine predicted posterior probabilities for class membership than class predictions.

Caruana et al. [5] evaluated stacked generalization with logistic regression with thousands of classifiers on binary classification problems, and reported that stacked generalization tended to overfit, resulting in poor overall performance. In this paper, we remedy this overfitting and improve overall generalization accuracy through regularization.

[1] Wolpert introduced the ideas of internal cross-validation and trainable combiner functions together in his article; we use the term 'stacked generalization' to refer to the latter.

Regularization attempts to improve predictive accuracy by reducing variance error at the cost of slightly increased bias error–this is known as the bias-variance tradeoff [8]. In this paper, regularization is applied to linear stacked generalization for multi-class classification in order to improve predictive accuracy. In particular, Ridge regression [8], lasso regression [8], and elastic net regression [9] are used to regularize the regression model by shrinking the model parameters. Lasso regression and some settings of elastic net regression generate sparse models, selecting many of the weights to be zero. This means each class prediction may be produced by a different subset of base classifiers.

In our experiments, many classification algorithms and many parameter settings are used to build a library of base models as in Caruana et al. [10]. We also perform resampling at the ensemble level in order to obtain more statistically reliable estimates of performance without the expense of retraining base classifiers. We look at the correspondence between performance on subproblems and overall classifier performance, and interpret the behavior of sparse linear models in stacked generalization.

This paper is organized as follows: Section 2.1 formally describes stacked generalization, including usage of indicator functions to transform the multi-class problem into several regression problems and the class-conscious extension, StackingC. Section 2.2 describes linear regression, Ridge regression, lasso regression and elastic net regression, which are used to solve the indicator subproblems in stacked generalization. Section 3 describes empirical studies that indicate the advantage of regularization. Section 4 discusses the results and Section 5 concludes with a summary and future work.

2 Model

2.1 Stacked Generalization

Given a set of L classifiers $\hat{y}_i(\boldsymbol{x}|\boldsymbol{\theta})$, $i = 1..L$, the predictions of each classifier on a validation dataset $\mathcal{D}_{val}$ are aggregated and combined with the known labels to create a meta-level training dataset $\mathcal{D}'_{val}$. The combiner function is then trained on this meta-level validation dataset. Given a test point, the predictions of all base-level classifiers are combined to produce a new data point $\boldsymbol{x}'_i$. The combiner function is evaluated at the new data point $\boldsymbol{x}'_i$, and its output is taken as the ensemble output. Formally, the ensemble prediction of stacked generalization is given by $sg(\boldsymbol{x}) = c(y_{11}(\boldsymbol{x}), ..., y_{1K}(\boldsymbol{x}), ..., y_{L1}(\boldsymbol{x}), ..., y_{LK}(\boldsymbol{x}))$, where $\boldsymbol{x}$ is the test point, c is the classifier combiner function and y_{lk} is the posterior prediction of the l^{th} classifier on the k^{th} class. Following Ting and Witten, a regression function can be used at the meta-level by constructing one regression subproblem per class with an indicator function [7]. At prediction time, the class corresponding to the subproblem model with the highest output is taken as the ensemble output. A more general discussion of reducing classification to linear regression problems is given in Hastie et al. [8].

The most general form of stacked generalization includes all outputs from all base classifiers. To simplify the problem, Seewald recommends using a class-conscious approach in which each indicator model is trained using predictions on the indicated class only, called StackingC [11]. Formally, the StackingC class prediction is given by $sc(\boldsymbol{x}) = \operatorname{argmax}_k r_k(y_{1k}(\boldsymbol{x}), ..., y_{Lk}(\boldsymbol{x}))$, where $\boldsymbol{x}$ is the test point, $k = 1..K$ is an index over classes, r_k is the regression model for the indicator problem corresponding to the k^{th} class and y_{lk} is the posterior prediction of the l^{th} classifier for the k^{th} class. Ting and Witten report that StackingC gives comparable predictive accuracy while running considerably faster than stacked generalization [7], and Seewald also reports increased predictive accuracy. Based on these arguments, the experiments in this paper use StackingC rather than complete stacked generalization.

2.2 Linear Models and Regularization

The least squares solution is given by $\hat{y} = \hat{\boldsymbol{\beta}}\boldsymbol{x}$, where $\hat{\boldsymbol{\beta}} = \operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$. Here $\hat{\boldsymbol{\beta}}$ is the vector of model parameters determined by the regression, N is the number of training data points, y_i is the true output on data point i, x_{ij} is the jth feature of the ith data point, and p is the number of input dimensions for the problem. In StackingC, the features are predicted probabilities from base classifiers. When the linear regression problem is underdetermined, there are many possible solutions. This can occur when the dimensionality of the meta-feature space L is larger than the effective rank of the input matrix (at most N), where L is the number of classifiers and N is the number of training points. In this case, it is possible to choose a basic solution, which has at most m nonzero components, where m is the effective rank of the input matrix.

Ridge regression augments the linear least squares problem with an L2-norm constraint: $P_R = \sum_{j=1}^{p} \beta_j^2 \leq s$. This has the effect of conditioning the matrix inversion problem by adding a constant k to the diagonal: $\boldsymbol{\beta} = (X^TX + kI)^{-1}X^T\boldsymbol{y}$. There is a one-to-one correspondence between s and k [8].

Lasso regression augments the linear least squares problem with an L1-norm constraint: $P_l = \sum_{j=1}^{p} |\beta_j| \leq t$. The L1-norm constraint makes the optimization problem nonlinear in y_i, and quadratic programming is typically used to solve the problem. Unlike Ridge regression, lasso regression tends to force some model parameters to be identically zero if the constraint t is tight enough, thus resulting in sparse solutions.

Zou and Hastie describe a convex combination of the Ridge and lasso penalties called the elastic net [9]. The penalty term is given by $P_{en}(\boldsymbol{\beta}|\alpha) = (1 - \alpha)\frac{1}{2}||\boldsymbol{\beta}||_{l2}^2 + \alpha||\boldsymbol{\beta}||_{l1}$, where $0 \leq \alpha \leq 1$ controls the amount of sparsity. The elastic net is particularly effective when the number of predictors p (or classifiers in StackingC) is larger than the number of training points n. The elastic net performs groupwise selection when there are many correlated features (unlike the lasso, which instead tends to select a single feature under the same circumstances). When there are many excellent classifiers to combine, their outputs will be highly correlated, and the elastic-net will be able to perform groupwise selection.

3 Experimental Studies

For empirical evaluations, we selected publicly available datasets with numerical attributes and $k \geq 3$ classes. Table 1 indicates the datasets and relevant properties. For the 26-class letter dataset, we randomly subsampled a stratified selection of 4000 points.

Table 1. Datasets and their properties

Dataset	*Attributes*	*Instances*	*Classes*
balance-scale	4	625	3
glass	9	214	6
letter	16	4000	26
mfeat-morphological	6	2000	10
optdigits	64	5620	10
sat-image	36	6435	6
segment	19	2310	7
vehicle	18	846	4
waveform-5000	40	5000	3
yeast	8	1484	10

Approximately half the data points (with stratified samples) in each problem are used for training the base classifiers. The remaining data is split into approximately equal disjoint segments for model selection at the ensemble level (e.g. stacking training data or select-best data) and test data, again in stratified samples. In a real-world application, the base classifiers would be re-trained using the combination of base-level data and validation data once ensemble-level hyperparameters are determined, but this is not done in our studies due to the expense of model library construction.

Previous studies with $L \geq 1000$ classifiers obtain one sample per problem, with no resampling due to the expense of model library creation, such as in Caruana et al. [5]; we partially overcome this problem by resampling at the ensemble training stages. In particular, we use Dietterich's 5x2 cross-validation resampling [12] over the ensemble training data and test data. We use the Wilcoxon signed-rank test for identifying statistical significance of the results, since the accuracies are unlikely to be normally distributed [13].

We generate around 1000 classifiers, including neural networks, support vector machines, k-nearest neighbors, decision stumps, decision trees, random forests, and adaboost.m1 and bagging models. See the Appendix for full details on construction of base classifiers. Source code and data sets are available at `http://spot.colorado.edu/~reids/reg-09/`

3.1 Ensemble Techniques

As a baseline for comparison, we select the best classifier as identified by accuracy on the held-out ensemble training set (*select-best*). We also compare our linear models to voting (*vote*) and averaging (*average*) techniques. The StackingC approaches are denoted *sg-linear*, *sg-ridge* and *sg-lasso*.

For the majority of our datasets, there are more linear regression attributes p (same as the number of classifiers L) than data points n (equal to the number of stacking training points, roughly $\frac{N}{4}$)[2]. To solve this underdetermined system without resorting to the typical Ridge solution, we choose a basic solution as implemented in the Matlab *mldivide* function.

In order to select the Ridge regression penalty, we search over a coarse grid of $\lambda = \{0.0001, 0.01, 0.1, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\}$ using cross-validation, then use all validation data to train the ridge regression model with the selected penalty parameter. We use the Matlab implementation of Ridge regression from the Matlab Statistics Toolbox. Parameters are selected by cross-validation for each subproblem rather than choosing a single λ for all subproblems. For example, the regularization hyperparameter for the first indicator problem λ_1 may differ from λ_2. For lasso regression, we use the LARS software by Efron and Hastie [14], and search over a grid of $fraction = 0$ to 1 in increments of 0.01 to select the regularization penalty term by cross-validation for each subproblem. We search over a finer grid in *sg-lasso* than in *sg-ridge* since model selection is much more efficient in LARS. For the elastic net, we use the glmnet package written by Friedman, Hastie and Tibshirani and described in the corresponding technical report [15].

4 Results

The test set accuracies of all ensemble methods are shown in Table 2. Each entry in this table is an average over 10 folds of Dietterich's 5x2 cross-validation [12] over ensemble training/validation data. According to the pairwise Wilcoxon

Table 2. Accuracy of each model for each data set. Entries are averages over the 10 samples from Dietterich's 5x2 cross-validation at the ensemble level. Variances are omitted based on arguments in Demšar [13]. See Section 3 for a description of the methods and Section 4 for discussion.

Dataset	$select - best$	$vote$	$average$	$sg-linear$	$sg - lasso$	$sg - ridge$
balance-scale	**0.9872**	0.9234	0.9265	0.9399	0.9610	0.9796
glass	0.6689	0.5887	0.6167	0.5275	0.6429	**0.7271**
letter	0.8747	0.8400	0.8565	0.5787	0.6410	**0.9002**
mfeat-m	0.7426	0.7390	0.7320	0.4534	0.4712	**0.7670**
optdigits	0.9893	0.9847	0.9858	0.9851	0.9660	**0.9899**
sat-image	0.9140	0.8906	0.9024	0.8597	0.8940	**0.9257**
segment	0.9768	0.9567	0.9654	0.9176	0.6147	**0.9799**
vehicle	0.7905	0.7991	0.8133	0.6312	0.7716	**0.8142**
waveform	0.8534	0.8584	**0.8624**	0.7230	0.6263	0.8599
yeast	**0.6205**	0.6024	0.6105	0.2892	0.4218	0.5970

[2] The *waveform*, *letter*, *optdigits* and *sat-image* datasets are exceptions.

signed-ranks test [13], Ridge regression StackingC outperforms unregularized linear regression StackingC at $p \leq 0.002$. *Select-best* outperforms both unregularized and lasso regression StackingC at $p \leq 0.002$. Ridge regression StackingC outperforms *select-best* at $p \leq 0.084$, and has more wins than any other algorithm. On two problems, *select-best* outperforms all model combination methods. On all problems, *sg-linear* and *sg-lasso* perform less accurately than *sg-ridge*; this suggests that it may be more productive to assign nonzero weights to all posterior predictions when combining several base classifiers.

To study the effect of regularization on each subproblem, we plot the root mean squared error for a particular indicator subproblem as a function of the regularization penalty hyperparameter. Computation of a reasonable composite value over all data sets is difficult due to incommensurability of the problems, so we restrict our focus to a particular subproblem[3]. Figure 1(a) shows the root mean squared error in the first subproblem in the *sat-image* dataset[4]. As the ridge penalty λ increases from 10^{-8} to 10^3, the error decreases by more than 10%. With such a small penalty term, the error at 10^{-8} roughly corresponds to the error that would be obtained by unregularized linear regression. For individual subproblems, therefore, regularization dramatically improves performance.

Figure 1(b) shows the overall accuracy of the multi-response linear regression with Ridge regularization for the *sat-image* dataset. Regularization increases the accuracy of the overall model by about 6.5%, peaking around $\lambda = 10^3$. As the penalty is increased beyond 10^3 (not pictured), the accuracy decreases, reaching 0.24, the proportion of the predominant class, around $\lambda = 10^8$. Please note that in this figure, λ is the same for all subproblems.

Figure 1(c) shows the correlation between the accuracy of the overall multi-response linear regression system and the root mean squared error on the first subproblem. The fit is approximately linear, with $a = -0.408e + 0.957$, where a is the accuracy of the multiclass classifier and e is the RMSE of the classifier on the first indicator subproblem.

Figure 2(a) shows the overall accuracy of the multi-response linear regression system as a function of the penalty term for lasso regression for the *sat-image* problem. Standard errors over the 10 folds are indicated. As in the Ridge regression case, λ is the same over all subproblems in this figure. The accuracy falls dramatically as the penalty increases beyond 0.2, stabilizing after $\lambda = 0.50$ at an accuracy of 0.24, the proportion of the predominant class.

In order to view the effect of the elastic net's mixing parameter α on the accuracy of the multi-response system, accuracy vs penalty curves are plotted in Figure 2(b) for $\alpha = \{0.05, 0.5, 0.95\}$. The $\alpha = 1.0$ curve indicated in Figure 2(a) is highly similar to the $\alpha = 0.95$ curve, and therefore omitted from Figure 2(b) for clarity. With a small penalty term $\lambda \leq 10^{-1}$, the curves are constant, and within one standard deviation of the *select-best* curve. As the penalty increases, the accuracy reaches a maximum that is dependent on α, with higher α values yielding

[3] Results are qualitatively similar for other subproblems.

[4] The root mean squared error is used instead of the accuracy because the subproblem in multi-response is a regression problem.

higher accuracy at smaller penalty values. In this case, fine-tuned regularization increases accuracy by about 1.5%.

In Ridge-regularized stacked generalization, all predictors are given some portion of the total weight. In lasso regression and some settings of elastic net regression, it is possible to obtain a sparse model in which many weights are identically zero. This reduces computational demand at prediction time and makes it

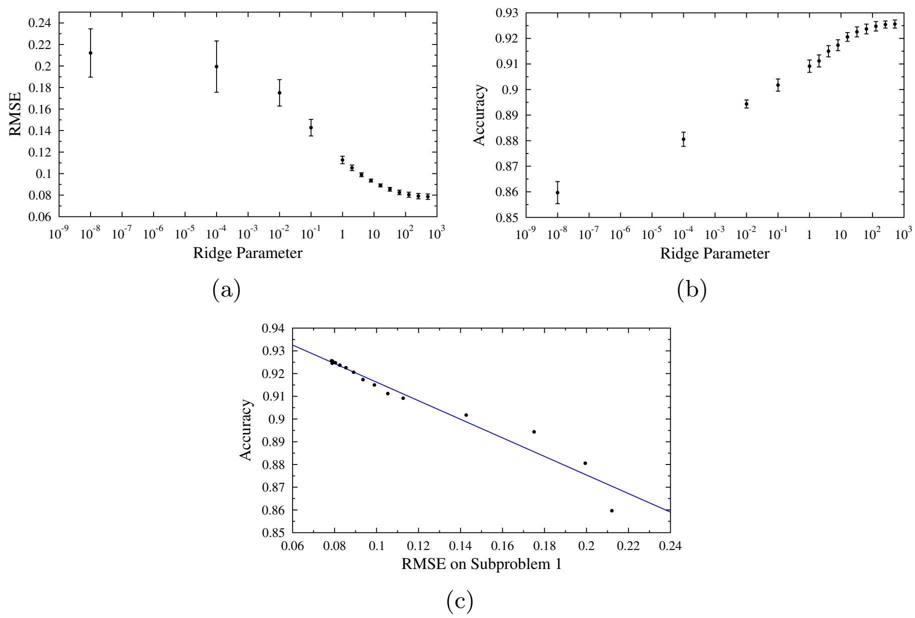


Fig. 1. Figure 1(a) shows root mean squared error for the indicator problem for the first class in the *sat-image* problem. Figure 1(b) shows overall accuracy as a function of the Ridge parameter for *sat-image*. The error bars indicate one standard deviation over the 10 samples of Dietterich's 5x2 cross validation. Figure 1(c) shows the accuracy of the multi-response linear regression system as a function of mean squared error on the first class indicator subproblem for Ridge regression.

Table 3. Selected posterior probabilities and corresponding weights for the *sat-image* problem for elastic net StackingC with $\alpha = 0.95$. Only the 6 models with highest total weights are shown here. *ann* indicates a single-hidden-layer neural network, and corresponding momentum, number of hidden units, and number of epochs in training.

$Classifier$	$class-1$	$class-2$	$class-3$	$class-4$	$class-5$	$class-6$	$total$
adaboost-500	0.063	0	0.014	0.000	0.0226	0	0.100
ann-0.5-32-1000	0	0	0.061	0.035	0	0.004	0.100
ann-0.5-16-500	0.039	0	0	0.018	0.009	0.034	0.101
ann-0.9-16-500	0.002	0.082	0	0	0.007	0.016	0.108
ann-0.5-32-500	0.000	0.075	0	0.100	0.027	0	0.111
knn-1	0	0	0.076	0.065	0.008	0.097	0.246

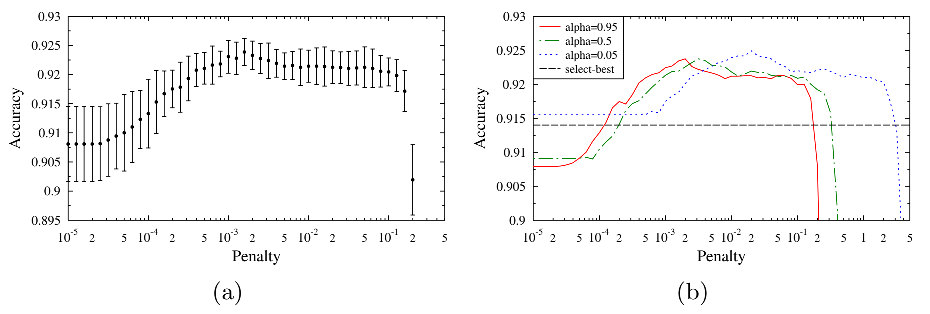


Fig. 2. Overall accuracy of the multi-response linear regression system as a function of the penalty term for lasso regression for the *sat-image* problem, with standard errors indicated in Figure 2(a). Figure 2(b) shows accuracy of the multi-response linear regression system as a function of the penalty term for $\alpha = \{0.05, 0.5, 0.95\}$ for the elastic net for *sat-image*. The constant line indicates the accuracy of the classifier chosen by *select-best*. Error bars have been omitted for clarity, but do not differ qualitatively from those shown in Figure 2(a).

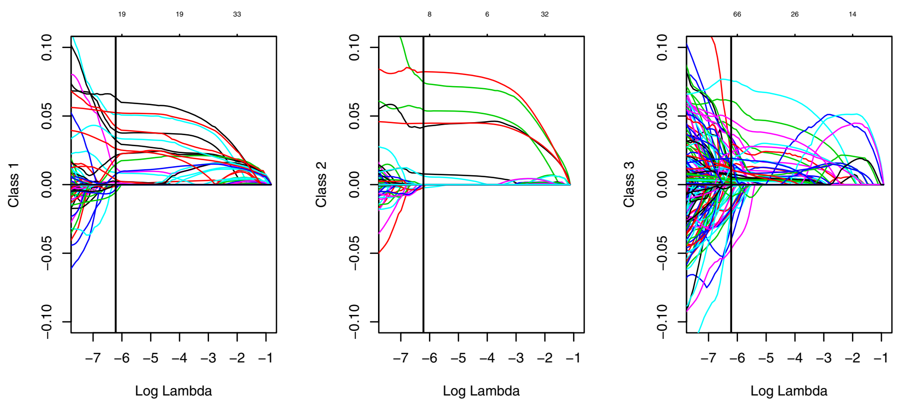


Fig. 3. Coefficient profiles for the first three subproblems in StackingC for the *sat-image* dataset with elastic net regression at $\alpha = 0.95$. This is over a single partition of ensemble training and testing data.

possible to identify a small subset of base classifiers and predictions that are responsible for making the overall prediction. Figure 3 shows the coefficient profiles for the *sat-image* dataset, for classes 1-3 with elastic net regularized StackingC with $\alpha = 0.95$. The optimal value of λ according to overall classification accuracy is shown as a vertical line. At $\lambda = \lambda_{opt}$, only 244 of the 999 classifiers are assigned weight for any of the subproblems. Table 3 shows the 6 classifiers with the highest total sum of weights for all classes. Sparse models obtained by L1-regularized linear regression can choose different classifiers for each class–that is, classwise posterior predictions are selected instead of complete classifiers. For instance, the classifier assigned the most total weight is $k = 1$-*nearest neighbor*, which contributes to the response for classes 3-6, but doesn't appear in the predictions for classes 1 or 2. The base classifier that makes the largest contribution to the *class-1* prediction is boosted decision trees run for 500 iterations. Thus each classifier is able to specialize in different class-based subproblems rather than being required to predict accurate probabilities for all classes.

5 Conclusion

Stacked generalization has a tendency to overfit; overfitting is even more likely when using many highly correlated, well-tuned models. In order to avoid overfitting and to improve prediction accuracy, it is necessary to perform regularization at the combiner level, even when using a linear combiner. Regularization can be performed by penalization of the L2 norm of the weights (Ridge regression), L1 norm of the weights (lasso regression) or a combination of the two (elastic net regression). L1 penalties yield sparse linear models; in stacked generalization, this means selecting from a small number of classifier posterior predictions.

An interesting extension of this work would be to examine the full Bayesian solutions (under Gaussian and Laplacian priors for regularization), instead of the single-point maximum likelihood estimates implicit in the Ridge (Gaussian prior) and lasso (Laplacian prior) regularizers. Other work could study additional regularization by (a) selecting a single regularization hyperparameter for use in all subproblems or (b) constraining the weights to be non-negative for each subproblem.

Acknowledgments

The authors would like to thank PhET Interactive Simulations at the University of Colorado at Boulder for their support, the Turing Institute, Glascow, Scotland and the UCI Repository for providing datasets and the reviewers for their helpful remarks.

References

1. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
2. Roli, F., Giacinto, G., Vernazza, G.: Methods for designing multiple classifier systems. In: Kittler, J., Roli, F. (eds.) MCS 2001. LNCS, vol. 2096, pp. 78–87. Springer, Heidelberg (2001)
3. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
4. Breiman, L.: Random forests. Mach. Learn. 45, 5–32 (2001)
5. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: ICML 2004: Proceedings of the twenty-first international conference on Machine learning, p. 18. ACM, New York (2004)
6. Wolpert, D.H.: Stacked generalization. Neural Netw. 5, 241–259 (1992)

7. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research 10, 271–289 (1999)
8. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, Heidelberg (2003)
9. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society B 67, 301–320 (2005)
10. Caruana, R., Munson, A., Niculescu-Mizil, A.: Getting the most out of ensemble selection. In: ICDM 2006: Proceedings of the Sixth International Conference on Data Mining, Washington, DC, USA, pp. 828–833. IEEE Computer Society, Los Alamitos (2006)
11. Seewald, A.K.: How to make stacking better and faster while also taking care of an unknown weakness. In: ICML 2002: Proceedings of the Nineteenth International Conference on Machine Learning, pp. 554–561. Morgan Kaufmann Publishers Inc., San Francisco (2002)
12. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation 10, 1895–1923 (1998)
13. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
14. Efron, B., Hastie, T., Johnstone, L., Tibshirani, R.: Least angle regression. Annals of Statistics 32, 407–499 (2004)
15. Friedman, J., Hastie, T., Tibshirani, R.: Regularized paths for generalized linear models via coordinate descent. Technical report, Stanford (2008)

Appendix: Base Classifiers

We generate about 1000 classifiers for each problem. For each classification algorithm, we generate a classifier for each combination of the parameters specified below. All implementations are in Weka except for the Random Forest (R), for which we used the R port of the Breiman-Cutler code by Andy Liaw, available through CRAN.

1. Neural Network decay={true, false} momentum={0.1, 0.5, 0.9} learningRate={0.5, 0.75, 0.9} trainingTime={100, 500, 1000} numHiddens={2, 4, 16, 32}
2. Support Vector Machine (C-SVM) kernelType={linear, polynomial, rbf, sigmoid} coef0={-1, 1} cost={0.1, 1.0, 10, 100, 1000} degree={1, 2, 3} eps={0.001, 0.01} gamma={0.1, 0.3, 0.8}
3. K-Nearest Neighbor k={1, 2, 4, 16, 32, 64}
4. Decision Stump
5. Decision Tree (J48) binarySplits={true, false} confidenceFactor={0.25, 0.5, 0.75} reducedErrorPruning={false, true} unpruned={true, false}
6. Random Forest (Weka) numTrees={1, 2, 30, 50, 100, 300, 500}
7. AdaBoost.M1 numIterations={10, 50, 100, 500} classifier={J48 binary Splits={true, false}, Decision Stump}
8. Bagging classifier={J48 binarySplits={true,false}} numBags={5, 10, 50}
9. Random Forest (R) numTrees={1, 2, 30, 50, 100, 300, 500}

Active Grading Ensembles for Learning Visual Quality Control from Multiple Humans

Davy Sannen and Hendrik Van Brussel

Katholieke Universiteit Leuven, Department of Mechanical Engineering
Celestijnenlaan 300B, B-3001 Heverlee (Leuven), Belgium
{Davy.Sannen,Hendrik.VanBrussel}@mech.kuleuven.be

Abstract. When applying Machine Learning technology to real-world applications, such as visual quality inspection, several practical issues need to be taken care of. One problem is posed by the reality that usually there are multiple human operators doing the inspection, who will inevitable contradict each other occasionally. In this paper a framework is proposed which is able to deal with this issue, based on trained ensembles of classifiers. Most ensemble techniques have however difficulties learning in these circumstances. Therefore several novel enhancements to the *Grading* ensemble technique are proposed within this framework – called *Active Grading*. The Active Grading algorithm is evaluated on data obtained from a real-world industrial system for visual quality inspection of the printing of labels on CDs, which was labelled independently by four different human operators and their supervisor, and compared to the standard Grading algorithm and a range of other ensemble (classifier fusion) techniques.

Keywords: Ensemble learning, grading, classifier fusion, visual quality control, learning from multiple humans.

1 Introduction

The most effective and flexible way to reproduce the human cognitive abilities needed to automate the required complex decision tasks in production processes, such as visual quality inspection, is by learning these tasks from human experts [1]. Traditionally, this is done using supervised learning, the data for which is provided by one selected person. The learning system is trained on this single set of data items, each of which has a unique label assigned to it. There may be some minor inconsistencies within the data, but these are usually considered as being random and each label is considered to be the ground truth.

However, quality inspection systems nowadays require the highest possible flexibility (due to e.g. changing customer demands, slight changes in the production line, new products to be inspected, etc.) [2]. This requires the human operators, currently performing their task manually, to be able to directly train and adapt the system without too much intervention of their supervisor. A typical situation is that there are three shifts and one operator per shift is working on the system. Their supervisor would like to be in control of their decisions as much as possible, but does not perform the inspection him-/herself.

Visual quality inspection is difficult because it is based on human evaluations which cannot be converted (easily) into mathematical rules. The literature shows that the effectiveness of human visual quality inspection lies around 80% [3]. This means that in 20% of the cases the decision a human operator makes is different from his/her supervisor. These can be caused by a number of factors, such as different levels of experience and training or fatigue and stress, caused by the typically very strict time restrictions. Therefore techniques are needed which can deal with these contradictions and inconsistencies in a systematic way if we want the human operators to train the system themselves.

This paper proposes an approach for this kind of problem based on ensembles of classifiers. Each of the operators will train their own personal classifier, which are afterwards combined by an ensemble method, trained to represent the supervisor's decisions as well as possible. The main difficulty is that for a substantial part of the data (about 20%), systematically *none* of the operators agrees with their supervisor (and hence also not the decisions of the classifiers each of the operators trains). Most ensemble methods cannot cope well with such a setting. To solve this problem an extension of the *Grading* ensemble method [4] will be presented which is able to combine the decisions of the classifiers, trained by the different operators, in an appropriate way.

The remainder of this paper is organised as follows. A general framework for learning visual quality inspection from multiple humans is proposed in Section 2, in which each of the operators trains his/her own personal classifier, which are afterwards combined by an ensemble method. A novel ensemble method – called *Active Grading* – which is able to effectively combine the decisions of the different operators is formulated in Section 3 as a generalisation of the *Grading* ensemble method [4]. This ensemble method is able to learn in the setting of the application in this paper, i.e. when for a substantial part of the data none of the classifiers in the ensemble provides the correct classification. Experiments were done using real-world data obtained from an industrial visual quality control application for CD imprints, described in Section 4 together with the obtained results. Finally, a conclusion is formulated in Section 5.

2 Architecture

In Figure 1 a generic framework for learning visual quality inspection from multiple human operators is shown. Starting from the original image of the product which is to be inspected (left-hand side of the figure), a "deviation image" is calculated. The grey-level value of each pixel in this image correlates to the degree of deviation from the "optimal" image of the product. Usually the image is mostly black, with the potentially defective parts highlighted by non-black groups of pixels. The contrast image is used to eliminate application-specific elements from subsequent processing steps. From the contrast image Regions Of Interest (ROIs) are extracted. Essentially this is a grouping of the non-black

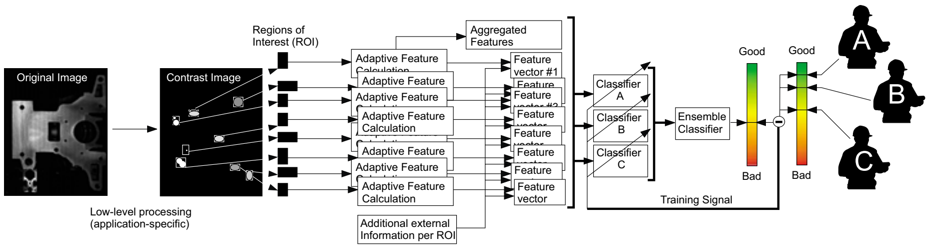


Fig. 1. General classification framework for visual quality inspection which can be trained by different human operators

pixels in the image into one or more distinct groups (called "objects"), each of which is a potential defect. The features of each object are calculated and can be complemented by three additional data sources: information about the ROIs, information about the status of the production process and aggregate features, characterising the images as a whole (information about all objects together) [5]. The feature vectors are processed by an operator-specific trainable classifier which generates a gradual good/bad decision for the entire image. This result is compared to the input of the human quality operator and a feedback loop, in the form of an (incremental) learning process, adapts the classification system. The operators, each training their own classifiers this way (during the initial training of the system or possible further adaptation), will inevitably provide different inputs to the system for some of the images – and thus their personal classifiers will also produce different classifications. These contradicting decisions are resolved using ensemble methods. This combination can be done using fixed rules or, if a supervisor labels the data as well, trainable ensemble methods, to better represent the decisions of the supervisor. The decision the ensemble makes is the final decision of the classification system.

Each of the operators will thus train their own personal classifier as they think would be best, according to their experience and expertise. Two levels of contradiction in the operators' decisions can be distinguished. The *inter-operator* contradictions are the *systematic* contradictions between the decisions of different operators. They can be caused e.g. by different levels of experience, training, skill, etc. The *intra-operator contradictions* are the contradictions an operator makes with decisions he has made himself. They can be caused by personal factors (such as the level of fatigue, attention, stress and boredom), environmental factors (such as a changed quality policy of the company and recent complaints of customers), etc. (see e.g. [6]).

The intra-operator contradictions, which are assumed to be basically random, are dealt with by the classifiers themselves. Several learning techniques can naturally handle noisy data (see e.g. [7]). The systematic inter-operator contradictions will be handled by the ensemble by combining the outputs of the different classifiers in a suitable and systematic way (see Section 3).

This architecture has several important advantages over other architectures which cannot deal with training input from different human operators. The classifiers trained independently by the operators will be easier to train, as they only need to handle the (non-systematic) intra-operator contradictions. The inter-operator contradictions are dealt with by the ensembles. Furthermore, it can be clearly distinguished what has been taught by which operator, enabling the system to give operator-specific feedback. The knowledge of each of the operators separately can be captured by the system.

3 The Active Grading Ensemble Framework

3.1 Combining the Decisions of Different Humans

There are two main requirements when selecting appropriate ensemble methods to be used in the architecture described in Section 2: (i) they have to combine an existing set of trained classifiers (trained by the operators); and (ii) the classifiers are no local experts in some parts of the feature space, but are trained over the entire features space (the classifiers are trained by the operators on the data provided by the inspection system, which cannot be influenced). The ensemble algorithms within the class of *classifier fusion* fulfill exactly these requirements. Classifier fusion techniques will be not explained in detail here; for reviews and detailed discussions see e.g. [8,9]. They will however be used for comparison in the evaluation in Section 4. Also another ensemble technique closely related to classifier fusion, called *Grading* [4], fits these requirements. This technique will be described in Section 3.2. To effectively tackle the problem in this paper the Grading algorithm will be reformulated in a novel "*Active Grading*" framework in Section 3.3, which is a generalisation of the original Grading technique. Within this framework, several enhancements to the original algorithm are described, which will enable the algorithm to learn from different humans.

3.2 Grading

Let us consider the case in which there is a diverse set of N^D trained classifiers available, $D_1, \ldots, D_{N^D}$, each trained to classify their own training data set into N^C different classes. Creating diversity in this set of classifiers can be done in different ways: different training samples can be selected to train the classifiers, different feature subsets can be selected, the target output can be changed, etc. (see e.g. [9,10]). Therefore, the training sets of the different classifiers can, but need not be of the same size or contain the same number of features. Note that in our application the diversity comes from changing the target output: each of the classifiers is trained by one of the human operators, which provides his own labelling for the training data. Assuming the training set of each of the classifiers D_i has a number of N_i^A attributes, the classifiers can be considered a mapping $D_i : \mathbb{R}^{N_i^A} \mapsto \mathbb{R}^{N^C}$, where the features are mapped to the classifier's confidence for each of the N^C different classes. Let us denote the confidence of classifier D_i

for class j when classifying a data item $\mathbf{x}$ as $D_{i,j}(\mathbf{x})$.[1] Without loss of generality we can assume $D_{i,j}(\mathbf{x}) \in [0, 1]$.

In the Grading ensemble method [4], for each of the (level-1) classifiers D_i a level-2 "*Grading classifier*" G_i is trained, which predicts whether or not the level-1 classifier will provide the correct prediction, based on the *original* classifier feature space. The Grading classifiers are thus mappings $G_i : \mathbb{R}^{N_i^A} \mapsto [0, 1]$, where 0 means the Grading classifier is perfectly sure the classifier will err; 1 means the Grading classifiers is perfectly sure the classifier will provide the correct classification. The training sets for the Grading classifiers are easily constructed by comparing the "crisp" classifier outputs with the target classifications. The crisp outputs of classifier D_i for a data item $\mathbf{x}$, $D_i^{\mathrm{Cr}}(\mathbf{x})$, can be obtained as follows:

$$\forall j : D_{i,j}^{\mathrm{Cr}}(\mathbf{x}) = \begin{cases} 1, \text{if } j = \arg\max_k D_{i,k}(\mathbf{x}); \\ 0, \text{else.} \end{cases} \tag{1}$$

Note that in the application in this paper, the target classifications are the labelling provided by the supervisor for (a part of) the training data.

When a new data item $\mathbf{x}$ is to be classified, each of the level-1 classifiers' predictions are obtained. The evaluation of the Grading classifiers is also obtained, indicating which of the level-1 classifiers is estimated to be correct. The final prediction of the Grading ensemble is obtained only from the level-1 classifiers which are estimated to be correct. If at least one classifier is estimated to be correct, the final prediction is calculated using the following formula [4]:

$$\forall j : \mathit{Grad}(\mathbf{x})_j = \sum_{i=1}^{N^D} \left\{ G_i(\mathbf{x}) | D_{i,j}^{\mathrm{Cr}}(\mathbf{x}) = 1 \wedge G_i(\mathbf{x}) > 0.5 \right\} \ , \tag{2}$$

where $\mathit{Grad}(\mathbf{x})_j$ is the final confidence of the Grading algorithm for class j.

If none of the classifiers is estimated to be correct the same procedure as above is applied, using $(1-G_i(\mathbf{x}))$ instead of $G_i(\mathbf{x})$ in (2). This comes down to using the classifications of all classifiers, even though they are estimated to be incorrect. In [4] this is described as being a "rare case" – although this may be true in the case of "standard" pattern recognition applications, this will not be the case in the application in this paper. The operators will contradict their supervisor *systematically* in some parts of the feature space, meaning that for significant parts of the feature space *none* of the classifiers (trained by the operators) will be correct (as will be shown in Section 4). Combining these outputs in the way the Grading algorithm does will result in incorrect classifications in these regions of the feature space. This is the problem the *Active Grading* approach will tackle, as will be explained in Section 3.3.

3.3 Active Grading

In this section the Grading algorithm [4] as described in Section 3.2 will be reformulated in a new "Active Grading" framework. Afterwards, enhancements

[1] $\mathbf{x}$ will be used to denote a data item, described by the appropriate features for the current classifier (if the classifiers are trained using different features).

to the Grading algorithm will be proposed within this framework, which will enable learning in the context of this paper – namely when none of the classifiers provide the correct classification for a significant part of the feature space.

Let us assume, like in Section 3.2, that we have a set of N^D trained classifiers, $D_1, \ldots, D_{N^D}$, and a set of N^D Grading classifiers, $G_1, \ldots, G_{N^D}$, each of which predicts whether its corresponding classifier will provide the correct classification for some new data item.

When a new data item $\mathbf{x}$ is to be classified, in the Active Grading framework a number of operations will be performed as follows. First, the classifier outputs are obtained and made crisp according to (1), resulting in $D_1^{\mathrm{Cr}}(\mathbf{x}), \ldots, D_{N^D}^{\mathrm{Cr}}(\mathbf{x})$. Also the outputs of the Grading classifiers are obtained (predicting whether the corresponding classifiers correctly classify $\mathbf{x}$), resulting in $G_1(\mathbf{x}), \ldots, G_{N^D}(\mathbf{x})$.

Next, for each classifier D_i, a *correction* operation is performed to correct the classifiers' outputs based on the outputs of the Grading classifiers, resulting in $D_i^{\mathrm{Corr}}(\mathbf{x})$. For the standard Grading algorithm described in Section 3.2 this operation is given by the following equation:

$$\forall i : D_i^{\mathrm{Corr}}(\mathbf{x}) = \begin{cases} G_i(\mathbf{x}) D_i^{\mathrm{Cr}}(\mathbf{x}), & \text{if } G_i(\mathbf{x}) > 0.5; \\ \left[1 - G_i(\mathbf{x})\right] D_i^{\mathrm{Cr}}(\mathbf{x}), & \text{else.} \end{cases} \tag{3}$$

Note that this operation is nothing more than a reweighing, which will be used further on in the case when none of the classifiers is estimated to be correct. At the end of this section an enhancement will be proposed which does perform a real correction of the classifier outputs, and which will form the heart of the Active Grading approach.

After the classifier outputs are "corrected", for each of the classifiers D_i an *inclusion* operation is performed to indicate which of the classifiers will participate in the final prediction, resulting in $I_i(\mathbf{x})$. For the standard Grading algorithm this operation is given by the following equation:

$$\forall i : I_i(\mathbf{x}) = \begin{cases} 0, \text{ if } G_i(\mathbf{x}) \leq 0.5 \wedge \exists k : G_k(\mathbf{x}) > 0.5; \\ 1, \text{ else.} \end{cases} \tag{4}$$

The final step in the Active Grading framework is the actual classifier fusion. The fusion of the Grading algorithm can now be simply written as follows:

$$\forall j : Grad_j(\mathbf{x}) = \sum_{i=1}^{N^D} \left\{ D_{i,j}^{\mathrm{Corr}}(\mathbf{x}) | I_i(\mathbf{x}) = 1 \right\} \ , \tag{5}$$

where $D_{i,j}^{\mathrm{Corr}}(\mathbf{x})$ denotes the confidence for class j of $D_i^{\mathrm{Corr}}(\mathbf{x})$ and $Grad_j(\mathbf{x})$ denotes the final predicted confidence of the Grading algorithm for class j.

The above formulation of the Grading algorithm does exactly the same thing as the algorithm described in Section 3.2. However, it provides a convenient framework for enhancements to this algorithm. As mentioned above, the main problem is how to handle situations in which none of the classifiers (are estimated to) provide the correct classification. Within the Active Grading framework introduced in this paper, we will propose an enhanced correction operation with respect to the one used in the standard Grading algorithm. When a classifier is estimated to be incorrect by its corresponding Grading classifier, we propose to effectively modify the output of the classifier in such a way that another class is predicted than the one which was initially predicted by the classifier. More formally, we propose to change (3) into the following equation:

$$\forall i : D_i^{\text{Corr}}(\mathbf{x}) = \begin{cases} G_i(\mathbf{x}) D_i^{\text{Cr}}(\mathbf{x}), & \text{if } G_i(\mathbf{x}) > 0.5; \\ [1 - G_i(\mathbf{x})]\left[1 - D_i^{\text{Cr}}(\mathbf{x})\right], & \text{else.} \end{cases} \tag{6}$$

Note that although at first glance this might seem to be a small modification, it has significant consequences. In the case that none of the classifiers is estimated to be correct, the standard Grading algorithm uses the (weighted) outputs of all of the classifiers without changing their "winning" classes. This will most likely provide incorrect classifications in this case. The Active Grading algorithm, however, effectively changes the "winning" classes the classifiers predict. By doing so, it actively uses the information provided by the Grading classifiers and modifies the classifier outputs accordingly. To the authors' knowledge, this is the first ensemble algorithm which changes the classifier outputs in such a way.

A second modification to the standard Grading algorithm is motivated by the idea that the corrected classifier outputs can be used, regardless whether the other classifiers are estimated to be correct or not. As the classifiers' outputs are corrected when the initial prediction of the classifiers is estimated to be incorrect, all classifiers can contribute valuable information to the ensemble. This can be very easily incorporated into the framework by using the following equation instead of (4): $\forall i : I_i(\mathbf{x}) = 1$. Intuitively, we estimate whether the classifiers are correct; if they are then their predictions are used, if they are not then their predictions are modified and these modified predictions are used.

4 Experimental Results

As discussed in Section 2, the proposed architecture for teaching the quality inspection to the system by multiple human quality control operators clearly has many advantages. By only taking into account the predictions of the operators' classifiers, we want to model the decisions of the supervisor. Of course, the accuracy of this system should not drop compared to a system in which one single classifier would only be trained on the data provided by the supervisor.

For the experiments in this paper a data set obtained from an industrial visual inspection system used for checking the quality of the labels printed on CDs is used. This data set contains 1534 samples and was independently labelled by 4 different operators and their supervisor into 2 classes: "good" and "bad". As discussed in Section 2, from the images obtained from the vision system 74 generic features (e.g. the number of objects detected, the area of the largest object, the maximum brightness of an object, etc.), describing each of the images, are derived [5]. Analysis of these data sets has shown that the operators make about the same decisions for the entire feature space, while in some part of the feature space (about 20% of the data) the supervisor makes different decisions

Table 1. Mean accuracy (in %) of CART classifiers for the CD data sets: the first row shows the evaluation of each of these classifiers for the operator's own (test) data; the second row shows the evaluation of these classifiers for the supervisor's data

Evaluation data	Training data provided by				
provided by	Operator01	Operator02	Operator03	Operator04	Supervisor
Same as training	94.77	96.60	97.39	96.01	94.38
Supervisor	75.16	70.00	73.86	73.66	94.38

Table 2. Mean accuracy (in %) of the different ensemble methods when combining the outputs of the classifiers, trained by the different operators, to model the supervisor's decisions for the CD data sets

Classifier fusion methods						Grading methods		
Vote	AC	FI	DT	DS	DDS	Grad	AGrad-N	AGrad-S
72.81	78.56	73.27	78.24	73.66	73.66	79.54	93.01	94.77

than *all* of the operators. Interestingly, this is about the error rate of human visual quality inspection reported elsewhere [3].

For the data sets provided by each of the operators a *CART* decision tree classifier [11] was trained. The accuracy of these classifiers was determined for all 5 data sets (using 10-fold cross-validation), the results of which can be found in the first row of Table 1. From these results it is clear that the classifiers are well trained for the data provided to them (ranging from 94.38% to 97.39%). However, when the operators' classifiers are evaluated on the data provided by the supervisor the accuracy drops significantly, ranging from 70% to 75.16% (the first 4 values in the second row of the Table 1), which are much lower than the 94.38% of the classifier trained by the supervisor himself. It is, however, the ensemble's job to combine the first four classifiers and to obtain an accuracy comparable to a classifier trained specifically on the supervisor's data.

The ensemble methods are trained on the same training data as the classifiers, so the outputs of the classifiers for their own training data are used as input to the ensembles. To combine the decisions of these classifiers the standard Grading [4] (*Grad*) and two variants of the proposed Active Grading approach are evaluated (Active Grading applied when none of the classifiers is estimated to be correct (*AGrad-N*) and applied for each of the classifiers separately when estimated to be incorrect (*AGrad-S*) – as detailed in Section 3.3). The *CART* decision tree classifier [11] was also used as Grading classifier. For comparison, a number of the most effective classifier fusion techniques (for detailed discussions, see e.g. [8,9]) were evaluated as well: Voting (*Vote*) [12], a number of simple Algebraic Connectives (*AC*) such as the Maximum, Minimum, Product, Mean and Median rules [13], Fuzzy Integral (*FI*) [14], Decision Templates (*DT*) [15], Dempster-Shafer combination (*DS*) [16] and its extension Discounted Dempster-Shafer combination (*DDS*) [17]. The results of these algorithms can be found in Table 2 (only the result of the best Algebraic Connective, the Product rule, is shown).

From the results in Table 2 it can be seen clearly that the classifier fusion algorithms do not perform well for this task. Their accuracies lie in the range of 72.81% to 78.56%. The standard Grading algorithm performs already slightly better with 79.54%, but this level of accuracy still is not enough for industrial applications. In contrast, the two proposed Active Grading approaches perform very well. *AGrad-N* and *AGrad-S* achieve accuracies of 93.01% and 94.77%, respectively. This means an improvement of 13.47% and 15.23% compared to the best of the other ensemble methods which were evaluated. It should be noted that the result of AGrad-S is even slightly better than a classifier trained specifically on the supervisor's data (see Table 1). This confirms that the decisions of the supervisor can effectively be modelled by the ensemble, if the classifiers trained by the different operators are combined in an appropriate way.

The reason the classifier fusion methods are not performing very well for this kind of problem is that they are trained on the *classifier outputs*, rather than on the original feature space. As for this application the majority of the data is correctly classified by the classifiers and the other part is *systematically* misclassified by each of the classifiers within the ensemble (trained by the operators), these methods will not be able to increase the performance of the system very much. In order to do this, information about the original feature space is required, which is used by the Grading methods. The standard Grading method can detect which of the classifiers will provide an incorrect prediction, but does not contain any mechanism to use this information in a constructive way. This is exactly what the Active Grading methods do: they actively modify the classifier outputs, so that they become useful for the combination process.

5 Conclusion

In this paper a framework for dealing with the reality that multiple human operators might be training a visual quality inspection system is proposed, in which the operators train their own personal classifier, the predictions of which are combined by an ensemble method. The operators' decisions are however not perfect and will systematically contradict their supervisor's decisions. This poses a problem for most ensemble techniques, which cannot cope well with the situation in which none of its member classifiers outputs the correct prediction. Therefore, the *Grading* ensemble technique is extended to a more general *Active Grading* framework, in which some extensions to the standard Grading method are proposed which make it able to learn in these circumstances. This technique is evaluated on data obtained from a real-world industrial system for visual quality inspection of the printing of labels on CDs, which was labelled independently by four different human operators and their supervisor, and compared to the standard Grading algorithm and a range of other classifier fusion algorithms. The experimental results show a performance boost of over 15% compared to the best other ensemble method.

Acknowledgments. This work was partly supported by the European Commission (project Contract No. STRP016429, acronym DynaVis). This publication reflects only the authors' view.

References

1. Castillo, E., Alvarez, E.: Expert Systems: Uncertainty and Learning. Springer, New York (2007)
2. Malamas, E., Petrakis, E., Zervakis, M., Petit, L., Legat, J.D.: A survey on industrial vision systems, applications and tools. Image and Vision Computing 21 (2003)
3. Juran, J., Gryna, F.: Juran's Quality Control Handbook, 4th edn. McGraw-Hill, New York (1988)
4. Seewald, A., Fürnkranz, J.: An evaluation of grading classifiers. In: Hoffmann, F., Adams, N., Fisher, D., Guimarães, G., Hand, D.J. (eds.) IDA 2001. LNCS, vol. 2189, pp. 115–124. Springer, Heidelberg (2001)
5. Sannen, D., Nuttin, M., Smith, J., Tahir, M.A., Caleb-Solly, P., Lughofer, E., Eitzinger, C.: An on-line interactive self-adaptive image classification framework. In: Gasteratos, A., Vincze, M., Tsotsos, J. (eds.) ICVS 2008. LNCS, vol. 5008, pp. 171–180. Springer, Heidelberg (2008)
6. Govindaraju, M., Pennathur, A., Mital, A.: Quality improvement in manufacturing through human performance enhancement. Integrated Manufacturing Systems 12(5) (2001)
7. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. John Wiley & Sons, New York (2000)
8. Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms. Wiley, Chichester (2004)
9. Polikar, R.: Ensemble based systems in decision making. IEEE Circuits and Systems Magazine 6(3), 21–45 (2006)
10. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: A survey and categorisation. Information Fusion 6(1), 5–20 (2005)
11. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth International Group, Belmont (1984)
12. Kuncheva, L., Whitaker, C., Shipp, C., Duin, R.: Limits on the majority vote accuracy in classifier fusion. Pattern Analysis & Applications 6(1), 22–31 (2003)
13. Kittler, J., Hatef, M., Duin, R., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
14. Cho, S., Kim, J.: Combining multiple neural networks by fuzzy integral for robust classification. IEEE Transactions on Systems, Man, and Cybernetics 25(2), 380–384 (1995)
15. Kuncheva, L., Bezdek, J., Duin, R.: Decision templates for multiple classifier fusion: An experimental comparison. Pattern Recognition 34(2), 299–314 (2001)
16. Rogova, G.: Combining the results of several neural network classifiers. Neural Networks 7(5), 777–781 (1994)
17. Sannen, D., Van Brussel, H., Nuttin, M.: Classifier fusion using Discounted Dempster-Shafer combination. In: Poster Proceedings of the 5th International Conference on Machine Learning and Data Mining, pp. 216–230 (2007)

Multiple Classifier Systems for Adversarial Classification Tasks

Battista Biggio, Giorgio Fumera, and Fabio Roli

Dept. of Electrical and Electronic Eng., Univ. of Cagliari
Piazza d'Armi, 09123 Cagliari, Italy
{battista.biggio,fumera,roli}@diee.unica.it
http://prag.diee.unica.it

Abstract. Pattern classification systems are currently used in security applications like intrusion detection in computer networks, spam filtering and biometric identity recognition. These are *adversarial* classification problems, since the classifier faces an intelligent adversary who adaptively modifies patterns (e.g., spam e-mails) to evade it. In these tasks the goal of a classifier is to attain both a high classification accuracy and a high *hardness of evasion*, but this issue has not been deeply investigated yet in the literature. We address it under the viewpoint of the choice of the architecture of a multiple classifier system. We propose a measure of the hardness of evasion of a classifier architecture, and give an analytical evaluation and comparison of an individual classifier and a classifier ensemble architecture. We finally report an experimental evaluation on a spam filtering task.

1 Introduction

Pattern recognition systems, and in particular multiple classifier systems, are currently used in several security applications like biometric identity recognition, intrusion detection in computer networks and spam filtering, in which the task is to discriminate "attack" samples (e.g., a spam e-mail) from "legitimate" samples (e.g., legitimate e-mails). These kinds of tasks are named *adversarial* classification problems, since there is an intelligent, adaptive adversary who tries to camouflage patterns (like spam e-mails) to evade the security system. Accordingly, in these applications the goal is to attain both a high classification accuracy and a high *hardness of evasion*, which is intuitively related to the effort required to the adversary to evade the system. However in the machine learning and pattern recognition literature the issue of the hardness of evasion in adversarial classification problems has not been deeply and formally investigated yet. Most of the works proposed countermeasures against specific kinds of attacks for spam filtering and intrusion detection tasks (see for instance [1,2,3]), and only few of them proposed formal models of adversarial classification tasks [4,5], or analysed the main issues raised by the application of machine learning techniques [6]. Therefore, from an engineering viewpoint the design of accurate and hard to evade classification systems for security applications is still an open problem.

In this work we argue that the hardness of evasion has to be taken into account in two distinct aspects of the design of a pattern recognition system: the choice of the features and the choice of the classifier architecture. Here we focus on the latter, and propose a quantitative measure of the hardness of evasion of a classifier architecture. We then analytically evaluate and compare the hardness of evasion and the accuracy of two specific single classifier and multiple classifier architectures which are used in many real security systems, and were supported so far only by intuitive arguments and empirical evidence. In light of our theoretical findings, we give an experimental evaluation of the accuracy and hardness of evasion of the considered classifier architectures on a spam filtering task, using the well known SpamAssassin open source spam filter.

2 Analysis of Multiple Classifier Systems for Adversarial Classification Tasks

In many classification systems used in security applications, like multimodal biometric authentication and verification, and intrusion detection in computer networks, the input features come from heterogeneous sources (for instance, images of faces and fingerprints). In these cases combining classifiers trained on the different feature subsets has been proposed as a natural way to design a simpler and more accurate classification system than a single classifier trained on all the available features [7,8,9,10]. Few authors proposed the use of MCSs with the explicit goal of improving the hardness of evasion (see for instance [2]). MCS architectures turn out to be used also in commercial and open source security systems, like the SpamAssassin spam filter (http://spamassassin.apache.org) and the Snort intrusion detection system (http://www.snort.org). However, with the only exception of a previous work by the authors [11], the use of MCSs for improving the hardness of evasion is supported only by intuitive and qualitative motivations, besides experimental evidences, and lack of a clear and sound theoretical support. In this section we propose a quantitative measure to evaluate the hardness of evasion of pattern classification systems, and apply it to analyse two different classifier architectures which are used in real adversarial classification tasks and are simple enough to allow for an analytical investigation.

2.1 The Concept of Hardness of Evasion

In security tasks there is a formal and agreed definition of classification accuracy in terms of the false positive (FP) and the false negative (FN) error rates. Instead, there is no formal and agreed definition of hardness of evasion. Intuitively, it depends on the "difficulty" for an adversary to evade the security system, but its evaluation depends on the specific task and on the kind of security system. Our aim is to propose a quantitative definition related to pattern classification systems. We first point out that in such systems the hardness of evasion can be analysed under two distinct aspects: the feature set, and the way in which features are combined (namely, the classifier architecture). Indeed, given an "attack" sample, an adversary has to consider two distinct issues: first, which features have to be modified, to evade the classifier? Second, how can patterns be camouflaged, so that the values of the targeted features are modified as desired? The latter issue is related to the nature of the individual features: the designer of the classifier should select features that are robust against pattern camouflage. However, in the design of a security system it is safer to assume that the adversary knows which features are used, and that he can always devise a way to evade them, although with some effort. Moreover, in practice quantifying the relative effort that is needed to modify different features is often very difficult. Accordingly, the hardness of evasion should also rely on forcing the adversary to modify as many features as possible to evade the system. This clearly depends on how the individual features are combined by the classifier architecture, which directly leads to the former issue above, namely, which (and how many) features have to be modified to evade the classifier. Accordingly, the hardness of evasion of a pattern classifier can be pursued at two distinct levels: the choice of the individual features, which should be not trivial to modify by pattern camouflage, and the choice of the classifier architecture, which should force the adversary to modify as many features as possible to evade the classifier. Although these choices are not necessarily independent on each other, they can nevertheless be addressed separately (perhaps in a closed-loop design cycle). In this work, we focus on the latter issue, namely designing a hard to evade classifier architecture in the sense defined above. To this aim, we give the following quantitative definition of the hardness of evasion of a classifier architecture:

> *For a given feature set, the hardness of evasion is defined as the expected value of the minimum number of features which have to be modified to evade the classifier.*

Accordingly, given two different classifiers A and B trained on the same feature set, A is harder to evade than B, if the expected minimum number of features that need to be modified to evade A is higher than the one needed to evade B.

2.2 A Theoretical Analysis of Multiple Classifier Systems for Adversarial Classification Tasks

In this section we focus on two classifier architectures (a single classifier and a MCS) used in multimodal biometric systems, in the SpamAssassin anti-spam filter, and in the Snort intrusion detection system. We will analytically evaluate and compare their hardness of evasion, defined as in Sect. 2.1, and classification accuracy. We first construct a model of the classification problem and of the two architectures, suitable to an analytical investigation. We consider n binary-valued features taking on the values 0 and 1, denoting respectively the absence and the presence of a given "attack" characteristic (as happens in Snort, while in SpamAssassin there are also features related to "legitimate" characteristics, which take on the values 0 and -1). The classifier architectures are shown in

Fig. 1. The first one is a "monolithic" classifier: a linear combination of the features with a decision threshold, as in SpamAssassin. A variant of this architecture is used by Snort: the logical OR between all features (viewed as boolean values), where 1 corresponds to *true* (accordingly, a pattern is labelled as "attack" if at least one feature detects an attack characteristic). The second one is an ensemble of classifiers trained on disjoint feature subsets, as in multimodal biometric systems [7,8]. To allow a direct comparison with the monolithic architecture, we consider an implementation in which the individual ensemble members are linear classifiers and are combined with the OR logical function. We denote the class labels as A ("attack") and L ("legitimate"), and the random feature vector as $X = (x_1, ..., x_n) \in \{0, 1\}^n$. To make an analytical evaluation possible, we assume that features are i.i.d. The (common) class-conditional distribution of each feature will be denoted as p_{1A}, p_{0A}, p_{1L} and p_{0L}, where $p_{1A} = P(X_i = 1|X \in A)$ for any $i = 1, ..., n$, and so on (obviously, $p_{1A} = 1 - p_{0A}$ and $p_{1L} = 1 - p_{0L}$). We also consider all the weights of the monolithic linear classifier to be identical. This is reasonable, given that all features are assumed to have the same discriminant capability. Without loosing generality, we normalise the weight values to 1 and consider only a variable threshold $t > 0$. The decision function $s_{\mathrm{M}}(x)$ of the monolithic classifier can then be written as follows (see Fig. 1, left):

$$s_{\mathrm{M}}(x) = \begin{cases} 1, \text{ if } \sum_{i=1}^{n} x_i - t \geq 0, \\ 0, \text{ otherwise } . \end{cases} \tag{1}$$

Note that also the OR decision function used by Snort can be written as (1), provided that $t \in (0, 1]$. These architectures can also be viewed as MCSs, if features are the decisions of individual classifiers. We also consider the weights of the individual classifiers of the MCS to be all identical and normalised to 1, and a common value also for the decision thresholds, denoted with t'. Assuming further that the n features are uniformly subdivided among the N classifiers (this requires n to be multiple of N), the decision function of the m-th individual linear classifier of the MCS (Fig. 1, right) can be written as:

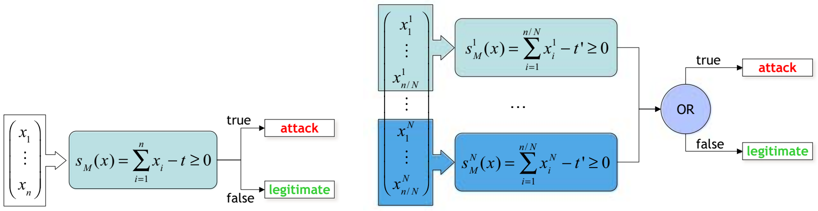
$$s_{\mathrm{M}}^{m}(x) = \begin{cases} 1, \text{ if } \sum_{i=1}^{n/N} x_i^m - t' \geq 0, \\ 0, \text{ otherwise } , \end{cases} \tag{2}$$

where x_i^m is the i-th feature of the m-th classifier. The MCS architecture is shown in Fig. 1, right.

We now compute the accuracy of the two classifiers above in terms of the FP and FN rates, as functions of n, N, t, t', and of the class-conditional feature distribution. The FP rate is the probability that a legitimate sample is misclassified as an attack, $FP = \mathrm{P}\left(s_{\mathrm{M}}(X) = 1|X \in L\right)$. For the monolithic classifier, from the definition of $s_{\mathrm{M}}(x)$ in (1), this happens if at least $\lceil t \rceil$ features equal 1 for a legitimate pattern. Being the features i.i.d., the corresponding probability is:

$$FP = \sum_{k=\lceil t \rceil}^{n} \binom{n}{k} p_{1L}^{k} \times p_{0L}^{n-k} \quad . \tag{3}$$

The FN rate equals 1 minus the true positive (TP) rate, which is defined as $\mathrm{P}\left(s_{\mathrm{M}}(X) = 1|X \in A\right)$. This equals the probability that at least $\lceil t \rceil$ features equal



$s_M(x) = \sum_{i=1}^{n} x_i - t \geq 0$
true
attack
false
legitimate
$s_M^1(x) = \sum_{i=1}^{n/N} x_i^1 - t' \geq 0$
$s_M^N(x) = \sum_{i=1}^{n/N} x_i^N - t' \geq 0$
OR
true
attack
false
legitimate

Fig. 1. The two classifier architectures considered in this work. A single, linear classifier (left), and an ensemble of linear classifiers combined by the OR logical function (right). In both cases the weights of the linear combination are assumed to be all identical, and a normalised value of 1 is considered.

1 for an attack pattern, and can be computed analogously to the FP rate. For the monolithic classifier one obtains:

$$TP = \sum_{k=\lceil t \rceil}^{n} \binom{n}{k} p_{1A}^{k} \times p_{0A}^{n-k} \quad . \tag{4}$$

Using the OR decision function instead of a linear combination, the expressions of FP and TP are the same ones above, with k ranging from 1 to n.

For the MCS, FP is the probability that at least one individual classifier outputs 1 for a legitimate sample. Each individual classifier is trained on n/N different i.i.d. features and has the same decision function (2). Their decisions are thus i.i.d. Denoting the common decision function as $s(x)$, one obtains:

$$\begin{aligned} FP &= \sum_{m=1}^{N} \binom{N}{m} \text{ P}\left(m \text{ classifiers say A } \wedge \ N-m \text{ say L} | X \in L\right) \\ &= \sum_{m=1}^{N} \binom{N}{m} \left[\text{P}\left(s(x)=1|X \in L\right)\right]^{m} \times \left[\text{P}\left(s(x)=0|X \in L\right)\right]^{N-m} \\ &= \sum_{m=1}^{N} \binom{N}{m} \left[\sum_{k=\lceil t' \rceil}^{n/N} \binom{n/N}{k} p_{1L}^{k} \times p_{0L}^{n/N-k}\right]^{m} \times \\ &\qquad \left[\sum_{k=n-\lceil t' \rceil}^{n/N} \binom{n/N}{k} p_{0L}^{k} \times p_{1L}^{n/N-k}\right]^{N-m} \quad . \end{aligned} \tag{5}$$

The TP rate of the MCS is the probability that at least one individual classifier outputs 1 for an attack sample. This can be computed analogously to (5):

$$\begin{aligned} TP = \sum_{m=1}^{N} \binom{N}{m} &\left[\sum_{k=\lceil t' \rceil}^{n/N} \binom{n/N}{k} p_{1A}^{k} \times p_{0A}^{n/N-k}\right]^{m} \times \\ &\left[\sum_{k=n-\lceil t' \rceil}^{n/N} \binom{n/N}{k} p_{0A}^{k} \times p_{1A}^{n/N-k}\right]^{N-m} \quad . \end{aligned} \tag{6}$$

The hardness of evasion was defined as the expected value over the distribution $\text{P}(X|X \in A)$ of the minimum number of features that have to be modified in an attack sample to evade the classifier. We denote with $n_{\min}(x)$ such value for any sample x, for the monolithic classifier (Fig. 1, left). Since x is labelled as A when at least $\lceil t \rceil$ features equal 1, denoting with $k(x)$ the number of features equal to 1 it follows that:

$$n_{\min}(x) = \begin{cases} k(x) - \lceil t \rceil + 1, & \text{if } k(x) \geq \lceil t \rceil, \\ 0, & \text{otherwise} \quad . \end{cases}$$

The expected value of $n_{\min}(x)$, denoted as $\overline{n}_{\min}$, can be computed as follows:

$$\begin{aligned}\overline{n}_{\min} &= \sum_{k=\lceil t\rceil}^{n} \left[k - \lceil t\rceil + 1\right] \times \mathrm{P}\left(k \text{ features equal } 1 | X \in A\right) \\ &= \sum_{k=\lceil t\rceil}^{n} \left[k - \lceil t\rceil + 1\right] \times \binom{n}{k} \times p_{1A}^{k} \times p_{0A}^{n-k} \ .\end{aligned} \tag{7}$$

To evade the MCS with the OR decision function (Fig. 1, right) it is necessary to evade all individual classifiers whose output is 1. Denoting with $n_{\min,m}(x)$ the minimum number of features that have to be modified in the m-th classifier, for a given attack sample x, the overall minimum number of features to modify is $n_{\min}(x) = \sum_{m=1}^{N} n_{\min,m}(x)$. The expectation is thus given by $\overline{n}_{\min} = \sum_{m=1}^{N} \mathrm{E}_{n_{\min,m}(X)|X\in A}\left[n_{\min,m}(X)\right]$. Since all classifiers are trained on disjoint subsets of i.i.d. features of the same size n/N and have the same decision function, the N random variables $n_{\min,m}(X), m = 1, ..., N$ are i.i.d. as well. Their expectation can be computed exactly as in (7), and thus we obtain:

$$\overline{n}_{\min} = N \times \sum_{k=\lceil t'\rceil}^{n/N} \left[k - \lceil t'\rceil + 1\right] \times \binom{n/N}{k} \times p_{1A}^{k} \times p_{0A}^{n/N-k} \ . \tag{8}$$

Since an analytical comparison between the above expressions of accuracy and hardness of evasion is not possible, we give a numerical comparison. To this aim, we first fix the class-conditional distribution of the features to values that can be representative of a real adversarial task like spam filtering (taking into account that features are assumed to be i.i.d.). We chose the values $p_{1A} = 0.25$ and $p_{1L} = 0.15$, namely, each individual feature detects 25% of the attacks and also erroneously identifies 15% of legitimate samples as attacks. We then evaluate the accuracy of the monolithic classifier and of the MCS using the receiver operating characteristic (ROC) curve (namely, the TP rate as a function of the FP rate, obtained by varying the decision thresholds t and t'). For the monolithic classifier (Fig. 2, left) we consider different values of the number of features n. As expected (being the features i.i.d.), the discriminant capability increases for increasing n. For the chosen values of p_{1A} and p_{1L}, $n = 600$ is sufficient to obtain nearly zero FP and FN rates. A realistic accuracy for spam filters is the one for n equal to about 300. The accuracy of the MCS was evaluated for different values of the ensemble size N, with n fixed to 300 (Fig. 2, right). It can be seen that the MCS discriminant capability is lower than that of the monolithic classifier (the MCS ROC curves are always below the one of the monolithic classifier for $n = 300$). The reason is that the individual classifiers of the MCS are much less accurate than the monolithic one, since they are trained on a lower number (n/N) of i.i.d. features. This turns out to be true also for different class-conditional feature distributions. We point out however that this result holds for the case in which the classifiers are *not* under attack.

We finally evaluate and compare the hardness of evasion (7) and (8) for $n = 300$ features. For a fair comparison between the monolithic classifier and the MCS we consider a fixed working point on the ROC curve defined by choosing classifier parameters (the decision thresholds t and t') that minimise a classification cost given by $FP + \frac{1}{C}FN$, where C denotes the relative cost of FP and FN errors. Since in security applications FP errors are more harmful than FN ones,

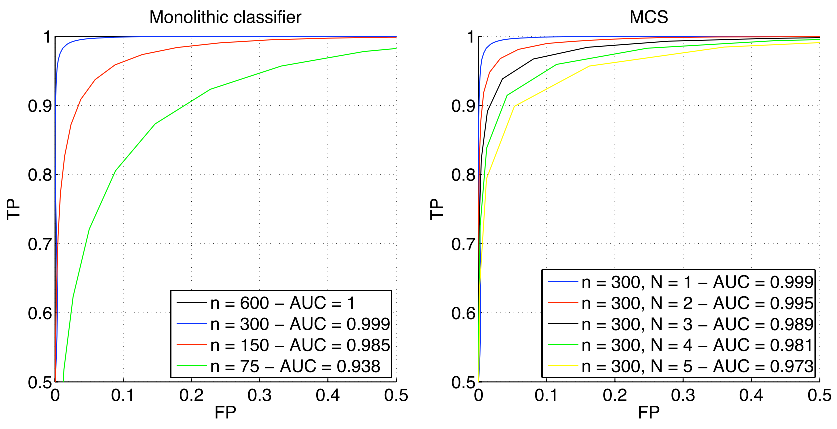


Fig. 2. Top-left part of the ROC curves of the monolithic linear classifier for different feature set sizes n (left), and of the MCS for $n = 300$ and different ensemble sizes N (right), for i.i.d. features with class-conditional distribution given by $p_{1A} = 0.25$ and $p_{1L} = 0.15$. The area under the ROC curve (AUC) is also reported.

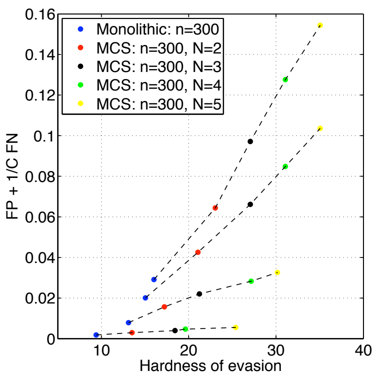


Fig. 3. Classification cost $FP + \frac{1}{C}FN$ as a function of the hardness of evasion for the monolithic classifier and four MCS with ensemble size $N = 2, ..., 5$, $n = 300$ features, and four C values. Each dashed line corresponds to a different C value: from top to bottom, $C = 1, 2, 10, 100$.

we consider $C > 1$. The comparison was made for four C values and four MCS ensemble sizes: $C = 1, 2, 10, 100$, and $N = 2, 3, 4, 5$. The corresponding classification cost and hardness of evasion are reported in Fig. 3. The comparison between the monolithic classifier and the MCSs, for any fixed C value, clearly shows that the monolithic classifier is more accurate at any given operating point when the adversary does not attack, but it is also easier to evade. Moreover, while the MCS accuracy decreases for increasing ensemble sizes, the hardness of evasion increases. Therefore, in the considered classifier architectures there is a trade-off between the accuracy when the classifier is not under attack, and the hardness of evasion. Note also that for increasing C values (namely, when a smaller FP rate is required), the accuracy of the MCS approaches that of the monolithic classifier, while the hardness of evasion remains significantly higher.

The analytical results in this section are limited to two classifier architectures, and hold only under rather strict conditions on the class-conditional feature distribution. Nevertheless, they allow to provide a first, formal evaluation and comparison of monolithic classifiers and MCSs in terms of both classification accuracy and hardness of evasion, and suggest that MCSs can be useful to attain a higher hardness of evasion than monolithic classifiers. In the next section we will give an empirical evaluation of these architectures for a spam filtering task, in light of the analytical results above.

3 Experimental Results

We analytically found in Sect. 2.2 that, when the adversary does not attack, a linear classifier with identical weights trained on i.i.d. features is more accurate than an ensemble of linear classifiers with identical weights and decision thresholds trained on disjoint subsets of identical size of the same features, and combined with the OR logical function, but it is easier to evade. In this section we empirically evaluate whether this result holds also in a real application where the assumption of i.i.d. features could be not satisfied. To this aim we considered the SpamAssassin spam filter (version 3.2.5), and the TREC 2007 e-mail corpus, publicly available at http://plg.uwaterloo.ca/~gvcormac/treccorpus07 and made up of 75,419 real e-mails (25,220 legitimate and 50,199 spam messages) collected between April 2007 and July 2007.

SpamAssassin can be considered as a linear classifier with several hundred binary features (rules associated to legitimate or spam e-mail characteristics take on respectively -1 and 0 values and 1 and 0 values), nine of which are actually associated to the outputs of a text classifier. The main aim of our experiments was to compare the two classification architectures of Fig. 1. To this end, we compared the SpamAssassin classifier architecture (a monolithic linear classifier) with MCSs trained on disjoint subsets of its features and combined with the OR logical function. However, even disregarding the nine tests associated to the text classifier, which exhibit a significantly higher discriminant capability, the remaining features cannot be considered i.i.d. Their correlation on the TREC legitimate e-mails ranges in $[-0.0045, 0.2821]$, with 0.0001 mean and 0.0025 std. dev., while for spam e-mails it ranges in $[-0.1867, 0.3265]$ with 0.0004 mean and 0.0069 std. dev. Their class-conditional distribution is given by $p_{1A} \in [0, 0.5588]$, with 0.0105 mean and 0.0374 std. dev., and by $p_{1L} \in [0, 0.0600]$ with 0.0003 mean and 0.0029 std. dev. To take this into account, we used different weights in the linear classifiers. Moreover, the text classifier of SpamAssassin (which has a continuous-valued output) was considered as one of the individual classifiers of the MCSs. The experiments were carried out as follows. We considered only the $n = 549$ features whose value was not constant over all e-mails of the TREC corpus. The text classifier was trained on the first $10,000$ e-mails in

Table 1. Mean and standard deviation of the FP and FN rates and of the hardness of evasion of the monolithic classifier trained on the SpamAssassin features, and of two MCSs with ensemble size $N = 3, 10$, trained on disjoint subsets of the same features

	FP rate	FN rate	hardness of evasion
monolithic	$0.0061(\pm 0.0024)$	$0.0363(\pm 0.0084)$	$1.37(\pm 0.09)$
MCS, $N = 3$	$0.0062(\pm 0.0013)$	$0.0520(\pm 0.0060)$	$3.01(\pm 0.13)$
MCS, $N = 10$	$0.0097(\pm 0.0020)$	$0.0569(\pm 0.0052)$	$3.25(\pm 0.22)$

chronological order. The next $10,000$ e-mails were used to train the linear classifiers, using a support vector machine (SVM) with the linear kernel (the publicly available `libsvm` software was used [12]). The operating point of all individual classifiers was set by keeping the FP rate below 1%. Two ensemble sizes for the MCS were considered: $N = 3, 10$. The 549 features were randomly and uniformly subdivided among the individual classifiers of the MCS. The accuracy (FP and FN rates) and the hardness of evasion at the chosen operating point were then computed on the remaining $55,419$ e-mails, and are reported in Table 1.

Table 1 shows that the considered MCS architecture provides a lower classification accuracy than the monolithic architecture, when they are not under attack (both the FP and FN rates of the MCS are slightly higher, and increase for increasing values of the ensemble size). However the hardness of evasion of the MCS is higher than the one of the monolithic classifier, and increases for increasing ensemble size. It is worth noting that this qualitative behaviour is the same found by our theoretical analysis of Sect. 2.2, although the assumption of i.i.d. features is violated, and the experimental setup does not match the one considered in Sect. 2.2 since the weights of the individual classifiers are not identical. In particular, the considered classifier architectures are characterised by a trade-off between classification accuracy and hardness of evasion: the MCS architecture can allow to improve the hardness of evasion, although its accuracy when the system is not under attack can be lower.

4 Conclusions

In this work we addressed for the first time the issue of quantitatively evaluating the hardness of evasion of a pattern classifier for security applications, and in particular of multiple classifier systems. We argued that the hardness of evasion has to be evaluated in two distinct steps of classifier design, namely the choice of the features and of the classifier architecture. We focused on the latter step, and proposed a quantitative measure of the hardness of evasion of a classifier architecture, related to the number of features that should be modified by the adversary to evade the whole classifier. This allowed us to give an analytical evaluation and comparison of two classifier architectures which are used in real security systems, but were motivated so far only by intuitive arguments and empirical evidence. The analytical results were exploited to give an experimental evaluation of these architectures in a real case study related to a spam filtering task. Our theoretical and experimental results suggest that MCSs can allow to improve the hardness of evasion, although their classification accuracy can be lower than that of a single classifier, when the system is not under attack. Moreover, the experimental results suggest that the validity of our theoretical conclusions can go beyond the assumptions under which they have been derived. We believe that the framework proposed in this work can be a starting point to derive principled guidelines for the design of pattern classifiers for adversarial classification problems.

References

1. Globerson, A., Roweis, S.T.: Nightmare at test time: robust learning by feature deletion. In: Cohen, W.W., Moore, A. (eds.) ICML. ACM International Conference Proceeding Series, vol. 148, pp. 353–360. ACM, New York (2006)
2. Perdisci, R., Gu, G., Lee, W.: Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In: International Conference on Data Mining (ICDM), pp. 488–498. IEEE Computer Society, Los Alamitos (2006)
3. Jorgensen, Z., Zhou, Y., Inge, M.: A multiple instance learning strategy for combating good word attacks on spam filters. Journal of Machine Learning Research 9, 1115–1146 (2008)
4. Lowd, D., Meek, C.: Adversarial learning. In: Press, A. (ed.) Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Chicago, IL (2005)
5. Dalvi, N., Domingos, P., Mausam, S.S., Verma, D.: Adversarial classification. In: Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Seattle, pp. 99–108 (2004)
6. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: ASIACCS 2006: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, pp. 16–25. ACM, New York (2006)
7. Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
8. Ross, A.A., Nandakumar, K., Jain, A.K.: Handbook of Multibiometrics. Springer Publishers, Heidelberg (2006)
9. Haindl, M., Kittler, J., Roli, F. (eds.): MCS 2007. LNCS, vol. 4472. Springer, Heidelberg (2007)
10. Giacinto, G., Roli, F., Didaci, L.: Fusion of multiple classifiers for intrusion detection in computer networks. Pattern Recognition Letters 24, 1795–1803 (2003)
11. Biggio, B., Fumera, G., Roli, F.: Evade hard multiple classifier systems. In: Okun, O., Valentini, G. (eds.) Supervised and Unsupervised Ensemble Methods and Their Applications. Studies in Computational Intelligence. Springer, Heidelberg (2009) (in press)
12. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm

Incremental Learning of Variable Rate Concept Drift

Ryan Elwell and Robi Polikar[*]

Signal Processing and Pattern Recognition Laboratory
Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028 USA
ryan.elwell@ieee.org, polikar@rowan.edu

Abstract. We have recently introduced an incremental learning algorithm, Learn^{++}.NSE, for *N*on-*S*tationary *E*nvironments, where the data distribution changes over time due to concept drift. Learn^{++}.NSE is an ensemble of classifiers approach, training a new classifier on each consecutive batch of data that become available, and combining them through an age-adjusted dynamic error based weighted majority voting. Prior work has shown the algorithm's ability to track gradually changing environments as well as its ability to retain former knowledge in cases of cyclical or recurring data by retaining and appropriately weighting all classifiers generated thus far. In this contribution, we extend the analysis of the algorithm to more challenging environments experiencing varying drift rates; but more importantly we present preliminary results on the ability of the algorithm to accommodate addition or subtraction of classes over time. Furthermore, we also present comparative results of a variation of the algorithm that employs an error-based pruning in cyclical environments.

Keywords: nonstationary environment, concept drift, Learn^{++}.NSE.

1 Introduction

Classification in changing environments is a particularly interesting and challenging problem with a growing list of application domains, such as network monitoring, economic, climate or financial data analysis, all of which generate data from such environments. The complexity of the problem is in part due to decision boundaries between classes being subject to potentially unpredictable change over time, which is commonly referred to as a non-stationary environment, and is most commonly associated with a change or *drift* in the underlying distribution of the data. The ability to track such data requires an algorithm that can update its parameters such that new knowledge is acquired, while old – and still relevant – information is retained, but the currently irrelevant information is discarded only to be recalled if such data later become relevant again in a cyclical environment. Ideally, such an algorithm should be incremental, i.e., not requiring access to previously seen data.

Earliest efforts in learning concept drift have focused on the types of drift and conditions under which such drift can be learned, e.g. under a formal PAC framework, as a hidden context [1;2], or more pragmatically as real vs. virtual drift [3]. More recently, Kuncheva indicated four different types of changes that may be encountered,

[*] Corresponding author.

such as noise (which should ideally be recognized as such, and ignored by the algorithm), a blip (a rare event, anomaly detection), an abrupt or gradual change [4;5].

Various approaches have been developed for concept drift, which we categorize as *active* or *passive* approaches, also referred to as *detect-and-retrain* and *constant update* [6]. An active approach explicitly seeks to detect drift in the data, and takes corrective action to keep the classifier up-to-date. FLORA is among the first such algorithms, and uses an adjustable window on the incoming data to ensure that only data from the current environment is used for classifier training [2]. The window size is based on a performance on current environment, which is also interpreted as a measure of the amount of drift. Any data that fall outside of the current window are then assumed to be irrelevant and the corresponding knowledge is forgotten by replacing the existing classifier with the newly trained one. In most active parameter or performance based drift detection approaches, the environment is assumed to be stationary unless some parameter (e.g. difference of certain statistical moments between prior and incoming data, [7-9] or performance threshold [10] is surpassed.

Passive approaches to tracking drift operate under the assumption that drift is always occurring, thus avoiding the complexity of trying to determine the magnitude of change. Therefore, the same procedure – e.g., using a fixed window size or always retraining a new classifier – is followed for each instance or batch of incoming data.

Ensemble or multiple classifier systems (MCS) based algorithms, which are typically passive approaches, represent a new breed of algorithms to learning in nonstationary environments, and they are particularly effective at providing a good balance between stability (retaining existing and relevant information) and plasticity (learning new knowledge) in the presence of drift. MCS-based approaches consist of an ensemble of classifiers, combined to form a final representative decision. In order to prevent irrelevant knowledge from effecting this decision, a combination of voting techniques and forgetting mechanisms are employed. Voting based classifier combination (ensemble weighting) allows classifiers with varying competences on the current environment to proportionally contribute to the final decision as in dynamic majority voting (DWM) [10], LIFT-based weight assigning [11], adaptive classifiers for changing environments (ACE) [12] or others [6;13]. Weights are often dynamically updated at each training instance, independent of the existence or amount of drift. While lower weights allow the knowledge of certain classifiers to be temporarily forgotten, *ensemble pruning* allows knowledge believed to be completely irrelevant to be permanently discarded.

Pruning can be useful because the ensemble, growing otherwise uncontrollably over time with new classifiers trained on new data, may accumulate too many irrelevant classifiers that can outvote the competent experts in the ensemble, despite weighting strategies. Pruning also helps with another concern with ensemble approaches, namely, the computational complexity that increases with each new classifier generated for incoming data. The preventative approach taken in many algorithms is therefore to limit the number of classifiers in the ensemble using an error-based criterion, where the worst performing expert in the ensemble is permanently removed. This has been shown to be an effective method in many studies, and has proven superior to the more basic age-based pruning (remove the oldest) [10;12-14]. We must note, however, that permanent removal of classifiers through pruning runs the risk of discarding information that may later become relevant, should the environment happen to be a cyclical one. The trade-off between using a pruning approach and retaining all classifiers is analyzed in this effort.

The aforementioned approaches have been evaluated on a variety of concept drift scenarios, such as gradual or abrupt, but not on cyclical environments, except our current and previous efforts [15-17], and certainly not on environments where the rate of change itself changes unpredictably, or where the change introduces or removes classes. In this contribution, we evaluate and compare our proposed approach, Learn^{++}.NSE, and its error-based pruning version on such challenging environments.

2 Learn^{++}.NSE

Learn^{++}.NSE is an ensemble-based (passive) approach for *N*on*S*tationary *E*nvironments. Data are drawn in batches over time from an environment that may be experiencing some change; however, we make no assumption on the nature or rate of the drift. The change – if and when it exists – may be gradual or abrupt, expanding or contracting in feature space, introducing or removing classes, cyclical or otherwise. We simply assume that the current data distribution has changed in some way compared to the prior distribution that provided the previous data. For each consecutive batch or snapshot of the data, a new classifier is generated. In the original version of Learn^{++}.NSE, all classifiers are retained, not only to maintain ensemble stability, but also to accommodate cyclic environments. To reduce outvoting from an eventually large number of potentially irrelevant classifiers, Learn^{++}.NSE tracks the age-weighted running average of the ensemble error over all current and previous environments, which are then used to determine classifier voting weights. A sigmoidal weighting gives most recent error a greater consideration for determining the classifier weights – regardless of the age of the classifier. Note that it is the error of the classifiers on more recent environments that are weighted more heavily, and not the classifiers themselves. Hence old classifiers may receive higher current weights.

The pseudocode in Figure 1 formally describes Learn^{++}NSE. Data are received as snapshots $\mathfrak{D}^t$ of the current environment with distribution $P^t(x,y)$. At each time step, m^tsamples are drawn as the training dataset $\{\boldsymbol{x}_i^t \in X; y_i^t \in Y\}, i = 1,\cdots,m^t$. A Base Classifier is used to train a classifier on the current dataset. Each consecutive snapshot $\mathfrak{D}^{t+1}$ is assumed to be drawn from distribution $P^{t+1}(x,y)$ that differs from that of the prior snapshot. Moreover, we assume that prior data is no longer accessible; thus the information must be stored solely within the classifiers. This assumption is a necessary condition for incremental learning algorithms.

When new data arrive at time t, Learn^{++}.NSE first evaluates the overall error ε_k^t of the existing ensemble of k classifiers on the new training data (Step 1). A normalized distribution D^t is adjusted – similar to that in AdaBoost [18] – according to the errors committed on each instance. This distribution is greater over those points at which the composite hypothesis $H^{t-1}(x_i)$ does not match the true class label (Step 2). This distribution D^t is an important tool to be used for individual classifier error evaluation. Once a new classifier is added to the ensemble (Step 3), the errors of all classifiers – normalized with respect to D^t – are computed on the new data $\mathfrak{D}^t$. Since the distribution D^t gives more emphasis to previously misclassified instances, the current error gives classifiers more credit for correctly classifying instances on which the ensemble hypothesis was incorrect. Note that if the newly trained classifier's error on the current environment exceeds ½, that classifier is discarded and a new one is trained; if the error of a previously generated classifier exceeds ½, however, its weight is set to ½. An error of ½ yields a normalized error β_k^t of 1 (Equation 6) at the current time step, which then carries zero voting weight (Equation 9).

The classifier errors over all (current and previous) environments are then averaged using a sigmoidal weighting function, giving more weight to errors on more recent environments (step 5). This allows retention and *reactivation* of old classifiers if they perform well on new environments. If the ensemble size exceeds a predetermined threshold *T*, and if pruning is desired, then the classifier with the largest error on the current data is discarded (step 6). The average error is then used to determine the voting weight of each classifier (step 8), combined through weighted majority voting (step 9).

```
Input: For each dataset $\mathfrak{D}^t$   $t = 1,2,\ldots$
Training data $\{\boldsymbol{x}_i^t \in X; y_i^t \in Y = \{1,\ldots,c\}\}, i = 1,\cdots,m^t$.
Supervised learning algorithm BaseClassifier
Ensemble size $T$
Do for $t = 1,2,\ldots$
  If $t = 1$, Initialize $D^1(i) = w^1(i) = 1/m^1$ , $\forall i$,  Go to step 3.  Endif    (1)
    1. Compute error of the existing ensemble on new data
       $E^t = \sum_{i=1}^{m^t}(1/m^t)\cdot [\![H^{t-1}(x_i) \neq y_i]\!]$    (2)
    2. Update and normalize instance weights
       $w_i^t = \frac{1}{m^t}\cdot\begin{cases} E^t, H^{t-1}(x_i) = y_i \\ 1, otherwise \end{cases}$,    (3)
       Set $D^t = w^t / \sum_{i=1}^{m^t} w_i^t \implies D^t$ is a distribution    (4)
    3. Call BaseClassifier with $\mathfrak{D}^t$, obtain $h_t: X \to Y$
    4. Evaluate all existing classifiers on new data $\mathfrak{D}^t$
       $\varepsilon_k^t = \sum_{i=1}^{m^t} D^t(i)\cdot [\![h_k(x_i) \neq y_i]\!]$ for $k = 1,\ldots,t$    (5)
       If $\varepsilon_{k=t}^t > 1/2$, generate a new $h_t$.  If $\varepsilon_{k<t}^t > 1/2$, set $\varepsilon_k^t = 1/2$,
       $\beta_k^t = \varepsilon_k^t/(1-\varepsilon_k^t)$, for $k = 1,\ldots,t$    (6)
    5. Compute weighted average of all normalized errors for $k^{th}$ classifier $h_k$:
       For $a,b \in \mathbb{R}$,  $\omega_k^t = 1/\left(1+e^{-a(t-k-b)}\right)$, $\omega_k^t = \omega_k^t / \sum_{j=0}^{t-k}\omega_k^{t-j}$    (7)
               $\bar{\beta}_k^t = \sum_{j=0}^{t-k}\omega_k^{t-j}\,\beta_k^{t-j}$, for $k = 1,\ldots,t$    (8)
    6. Ensemble Pruning (if used): If $t > T$
         Error-based: Remove $h_k$ where  $\varepsilon_k^t = \max_{k=1\ldots t}\varepsilon_k^t$
    Endif
    7. Calculate classifier voting weights  $W_k^t = \log(1/\bar{\beta}_k^t)$, $for\ k = 1,\ldots,t$    (9)
  → 8. Obtain final hypothesis: $H^t(x_i) = \arg\max_c \sum_k W_k^t \cdot [\![h_k(x_i) = c]\!]$    (10)
```

Fig. 1. Learn++.NSE Algorithm pseudocode

3 Experimental Results

Previous studies have shown that Learn^{++}.NSE i) consistently outperforms a single classifier, justifying an MCS approach; ii) is able to poll old classifiers in cases of recurring or cyclical data and consequently achieve higher accuracy; and iii) can work with a wide spectrum of on-line as well as batch learning algorithms, and hence is independent of the base classifier used to train the algorithm [15-17]. In this study, we pose the problem of even harsher environments which involve changes in the *number of classes* and changes in the *rate of drift* within the experiment (acceleration of drift). We also evaluate the trade-off made by pruning the ensemble and the ability to recall previously seen knowledge in recurring environments.

In prior work we have used a unique non-Gaussian dataset, the rotating checkerboard, derived from the canonical XOR problem. Figure 2 shows several snapshots of the distribution from which the data are drawn. A static window is maintained for sampling at each time step while the checkerboard itself rotates about a central axis. In each complete rotation ($\alpha = 0$ to 2π), the distribution inside the window repeats twice, yielding a recurring context. Random noise is introduced at each time step to prevent identical snapshots of training data from appearing. We also use a minimal number of training data (size m=25) to test the limit of the algorithm.

In this experiment, we compare the Learn^{++}.NSE performance during an accelerating (positive or negative) drift. We introduce three cases as described in Figure 3: a basic constant drift, an exponentially increasing drift, and a sinusoidal drift rate which includes a momentary pause at the midpoint of the test. From the start (t = 0) to finish (t=1), the board completes one rotation ($\alpha = 0$ to 2π) in T = 400 time steps, according the rate of changes shown in Fig. 3. The purpose of these tests is to investigate the algorithm's ability to track harsh environments while still maintaining the characteristics that have been realized thus far: the algorithms ability to significantly and consistently outperform a single classifier trained on the current training data (justifying an ensemble of classifiers), the ability to track recurring or cyclical data (justifying the error-weighting approach to prevent outvoting), and the ability to achieve a performance better than a pruned ensemble (justifying the retention of classifiers).

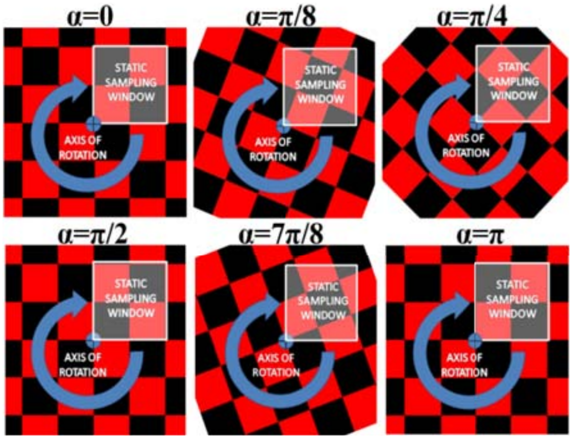


Fig. 2. Rotating checkerboard data

Results are computed on the entire testing grid of 51x51 resolution (2601 test points). All plots are averages of 100 independent trials and include a 95% confidence interval in a similarly colored shaded region around each curve (please see the electronic version of this paper for optimal viewing of the colors). For each experiment,

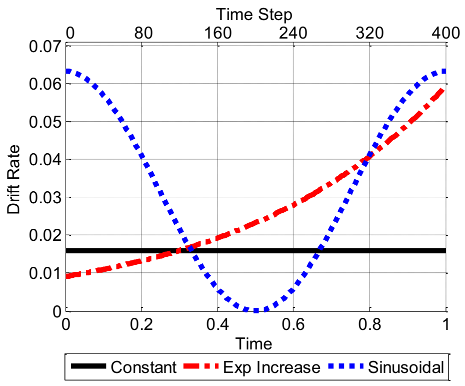


Fig. 3. Variable drift rates

we compare Learn^{++}.NSE to a single sifier and to Learn^{++}.NSE with error-based pruning, where a 25-classifier cap is maintained. Separate experiments are conducted for each dataset with different base classifiers including naive Bayes, MLP, and SVM. For brevity, and since all three yielded similar trends, only SVM results are given here.

Results in Figures 4 ~ 6 appear to be quite promising despite the changes in the amount and rate of drift throughout the experiment. First, Learn^{++}.NSE (with or without pruning) consistently outperforms a single classifier with statistical significance, regardless of the rate or type of the drift. This is not a trivial, but an important benchmark: a single classifier is always trained on the most recent data only, and never has to deal with former knowledge. The performance of Learn^{++}.NSE over a single classifier indicates that the algorithm can extract useful information from the previous classifiers, and that the ensemble structure is in fact beneficial. Second, the ability of the algorithm to track the changing the environment, as expected, is correlated to the current rate of change: when the environment is changing very slowly or when it is stationary, for example, during the mid sections of the sinusoidal rate of change (Fig. 6), the ensemble performance increases very rapidly. When the rate of change is accelerating (e.g. after 300 ms in Fig. 5, or 200 ms in Fig. 6), there is a slight drop in the performance; and when the environment is changing at a constant rate, the algorithm performance increases gradually (Fig 4.) We should note here that the sharp performance peaks in Fig 4 ~ 6 are simply due to the periodic nature of the problem, where decision boundaries become perpendicular (and therefore simpler) every $\pi/2$ radians. Also note that the current state of the board is different at any given time step for each of the experiments due to varying rates of change.

Perhaps one of the most interesting observations is the behaviour of the algorithm at and after $\alpha = \pi$. This happens at time step T = 200, 275 and 200 for the constant, accelerating and sinusoidal change of rate, respectively. As mentioned above, the distribution repeats itself in the $\alpha = [\pi \sim 2\pi]$ interval, creating a cyclic environment. Learn^{++}.NSE (both the pruned and unpruned versions) shows an increase in performance after this interval, compared to $\alpha = [0 \sim \pi]$ interval, a clear indication of the ability of the algorithm to use old classifiers which then become relevant again. This is particularly striking in Figure 5, where the performance improvement due to using old classifiers outweighs the performance drop due to rapidly accelerating rate of change which also occurs at around T =275 ($\alpha = \pi$). Finally, upon careful observations, we also observe that the unpruned Learn^{++}.NSE consistently and significantly outperforms the pruned version in the $\alpha = [\pi \sim 2\pi]$ interval (Fig. 4 and 5), indicating that those classifiers discarded by the pruning can in fact be useful in such a cyclic environment. The only exception to this is in Fig 6, where the large number of accumulated classifier during very slow / near stationary period of the sinusoidal rate does not allow the unpruned ensemble to significantly outperform the pruned ensemble for a short period of time (T=275) when the rate of change starts increasing again. It is interesting to note that during the very end (at around T = 400, or $\alpha = 2\pi$), both versions of Learn^{++}.NSE – but in particular the unpruned version – shows a more significant performance peak. This is attributed to the return of a prior distribution, now for the third time, allowing a larger number of classifiers to contribute to a correct decision.

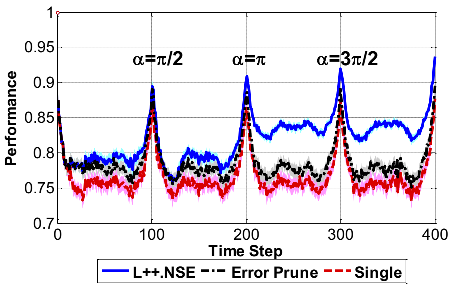


Fig. 4. SVM performance on checkerboard with constant drift rate

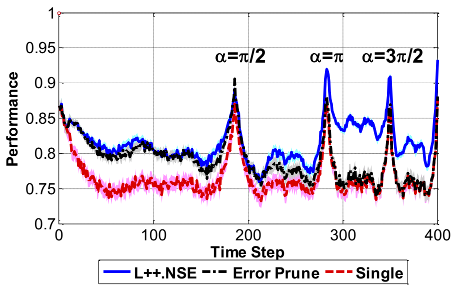


Fig. 5. SVM performance on checkerboard with exponentially accelerating drift rate

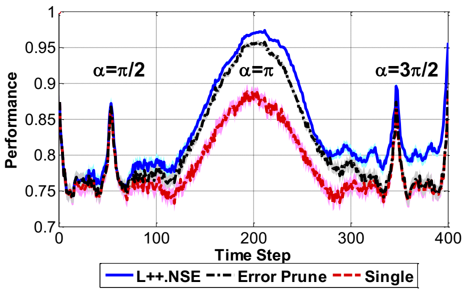


Fig. 6. SVM performance on checkerboard with sinusoidal drift rate

Hence, apart from validating that the algorithm can handle varying rates of drift, these experiments also allow us to determine when pruning may be beneficial: if the environment changes (relatively) at a constant rate, and/or the environment is not expected to repeat itself, then an error-based pruning provides performance nearly as good as the entire ensemble. However, in a cyclic environment (such as climate, electricity demand, etc. applications), using an unpruned ensemble is always preferable because of the sheer significance in performance increase.

The second experiment is a synthetic multi-class Gaussian dataset that includes random concept drift in the mean, variance, as well as the number of classes present at any given time. Figure 7 provides a visual display of the drift path followed by each class during the experiment, where the arrows indicate the direction of drift in mean for the next interval, and the sizes of the ellipses / circles indicate the variance of that class. Note that this scenario includes appearance of a new class (C_4) during

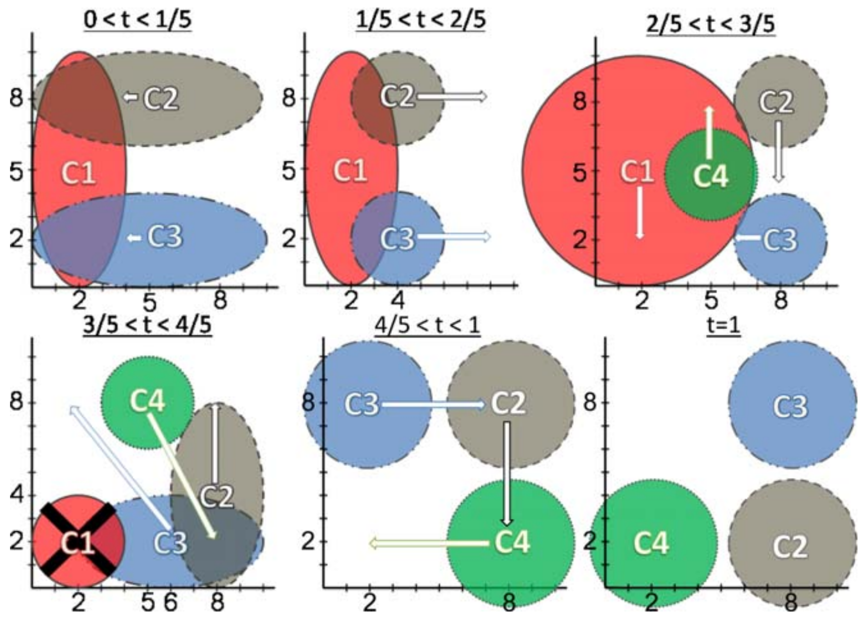


Fig. 7. Path of drift of multiclass Gaussian data from start at $t = 0$ to end at $t = 1$

$2/5 < t < 3/5$ and the disappearance of an old class (C_1) during $3/5 < t < 4/5$. At each time step in the $0 < t < 1$ interval, a new snapshot (size $m = 20$) of training data are obtained to train a new classifier for a total of T=300 time steps (snapshots).The performances of Learn^{++}.NSE (with and without pruning), a single classifier trained on the current training data, and that of ideal Bayes classifier (since the data distributions are Gaussian) are shown in Fig. 8 as averages of 100 independent trials along with their corresponding 95% confidence intervals (except for Bayes). The figure also indicates each separate interval of drift for reference.

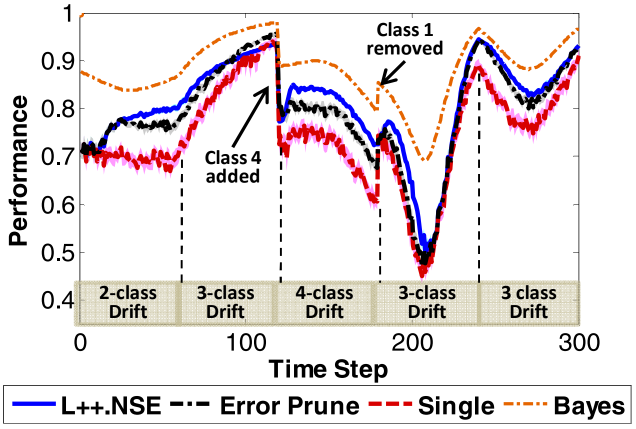


Fig. 8. Performance on the multiclass drift Gaussian data

The performance curve in Fig. 8 indicates that Learn^{++}.NSE and its pruned version are both capable of tracking an environment that experiences both concept drift and concept change (addition / removal of classes). The jumps in performance at T=120 and T=180 correspond to the sharp concept changes after the respective addition or removal of a class, which results (depending on the location of the class) in a change in the difficulty of the problem. We note that i) both versions of the algorithm follow the Bayes classifier very closely – hence able to track the drift, ii) there is little significant difference between the pruned and unpruned versions; and iii) they both outperform a single classifier, even (and especially) after class addition and subtraction.

4 Conclusions and Discussions

We presented an ensemble-based approach to the increasingly relevant topic of classification in non-stationary environments. We survey the complexity of this problem, as concept drift or change can occur in many forms (gradual, abrupt, cyclic, change in class counts, etc.) and rates (constant, increasing, decreasing, etc.). We show that Learn^{++}.NSE can closely follow and accommodate the drift, even in the harshest environments, regardless of its rate and type, or addition or removal of concept classes.

We have also compared the original version of Learn^{++}.NSE that retains all classifiers in case a cyclic environment makes the old classifier relevant to one that prunes the ensemble based on each classifier's error. The benefit of retaining all classifiers becomes especially evident when in fact an environment experiences a cyclic behavior, as the unpruned version significantly outperforms the pruned ensemble. However, in other cases, particularly when the rate of change is gradual and/or constant in rate, there is little or no significant difference between the two versions, and both versions significantly outperforms a single classifier trained on the most recent data.

Other attributes of the algorithm include independence of the base classifier being used, as well as being a truly incremental algorithm. At no point in time does Learn^{++}.NSE uses previously seen data and solely relies on the knowledge it extracts from the existing classifiers. Future work with Learn^{++}.NSE will include a comparison with other (active and passive) approaches on a variety of learning scenarios.

Acknowledgments

This material is based on work supported by the National Science Foundation under Grant No: ECS 0239090.

References

[1] Schlimmer, J.C., Granger, R.H.: Incremental Learning from Noisy Data. Machine Learning 1(3), 317–354 (1986)

[2] Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1), 69–101 (1996)

[3] Tsymbal, A.: Technical Report: The problem of concept drift: definitions and related work, Trinity College, Dublin, Ireland,TCD-CS-2004-15 (2004)

[4] Kuncheva, L.I.: Classifier Ensembles for Changing Environments. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 1–15. Springer, Heidelberg (2004)

[5] Kuncheva, L.I.: Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In: European Conference on Artificial Intelligence (ECAI), pp. 5–10 (2008)

[6] Rodriguez, J.J., Kuncheva, L.I.: Combining Online Classification Approaches for-Changing Environments. In: International Workshops on Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition S+SSPR (2008)

[7] Alippi, C., Roveri, M.: Just-in-Time Adaptive Classifiers;Part I: Detecting Nonsta-tionary Changes. IEEE Transactions on Neural Networks 19(7), 1145–1153 (2008)

[8] Da Silva, B.C., Basso, E.W., Bazzan, A.L.C., Engel, P.M.: Dealing with non-stationary environments using context detection. In: 23rd International Conference on Machine Learning - ICML 2006, vol. 2006, pp. 217–224 (2006)

[9] Oommen, B.J., Rueda, L.: Stochastic learning-based weak estimation of multino-mial random variables and its applications to pattern recognition in non-stationary environments. Pattern Recognition 39(3), 328–341 (2006)

[10] Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: an ensemble method for drifting concepts. Journal of Machine Learning Research 8, 2755–2790 (2007)

[11] Scholz, M., Klinkenberg, R.: Boosting Classifiers for Drifting Concepts. Intelligent Data Analysis, Special Issue on Knowledge Discovery from Data Streams 11(1), 3–28 (2007)

[12] Nishida, K., Yamauchi, K.: Adaptive Classifiers-Ensemble System for Tracking Concept Drift. In: 2007 International Conference on Machine Learning and Cybernetics, vol. 6, pp. 3607–3612 (2007)

[13] Wang, H., Fan, W., Yu, P., Han, J.: Mining concept-drifting data streams using en-semble classifiers. In: Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–235 (2003)

[14] Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: Seventh ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2001), pp. 377–382 (2001)

[15] Karnick, M., Muhlbaier, M.D., Polikar, R.: Incremental Learning in Non-Stationary Environments with Concept Drift Using a Multiple Classifier Based Approach. In: International Conference on Pattern Recognition (ICPR 2008), pp. 1–4 (2008)

[16] Karnick, M., Ahiskali, M., Muhlbaier, M.D., Polikar, R.: Learning concept drift in non-stationary environments using an ensemble of classifiers based approach. In: World Congress on Computational Intelligence, International Joint Conference on Neural Networks, pp. 3455–3462 (2008)

[17] Muhlbaier, M., Polikar, R.: An Ensemble Approach for Incremental Learning in Nonstationery Environments. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 490–500. Springer, Heidelberg (2007)

[18] Freund, Y., Schapire, R.E.: Decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)

Semi-supervised Co-update of Multiple Matchers

Luca Didaci[1], Gian Luca Marcialis[2], and Fabio Roli[2]

University of Cagliari
[1] Department of Pedagogical and Philosophical Sciences
[2] Department of Electrical and Electronic Engineering
Piazza d'Armi - 09213 Cagliari, Italy
{luca.didaci,marcialis,roli}@diee.unica.it

Abstract. Classification algorithms based on template matching are used in many applications (e.g., face recognition). Performances of template matching classifiers are obviously affected by the representativeness of available templates. In many real applications, such representativeness can substantially decrease over the time (e.g., due to "aging" effects in biometric applications). Among algorithms which have been recently proposed to deal with such issue, the template co-update algorithm uses the mutual help of two complementary template matchers to update the templates over the time in a semi-supervised way [8]. However, it can be shown that the template co-update algorithm derives from a more general framework which supports the use of more than two template matching classifiers. The aim of this paper is to point out this fact and propose the co-update of multiple matchers. Preliminary experimental results are shown to validate the proposed model.

1 Introduction

Template matching is a widely used classification approach in many applications [1-3]. For example, in personal verification systems based on multi-modal biometrics [2-4]. However, the effectiveness of these approaches is strongly dependent on the "representativeness" of templates. For example, in biometric applications, intra-class variations and aging (scratch on the finger skin or the face, unknown expressions, etc.) may affect such representativeness [2-3]. In order to deal with these problems, several "template update" algorithms have been proposed. Templates are usually captured under the supervision of a human expert (e.g. during the enrolment phase for biometric verification systems), but capturing all possible intra-class variations is impossible.

With regard to this issue, "semi-supervised" template update has been proposed [5-8]. In this approach, unlabelled samples (that is, samples collected during the system's operation) showing very strong similarity to existing templates are added into the set of samples of the related class, thus increasing the representativeness of related set of templates. Recently, the so-called "co-update" algorithm, derived from the co-training method [9], showed that the mutual "help" of two weakly correlated matchers can increase the template representativeness more quickly than systems adopting only one matcher, and improve the overall performance. This has been verified, in particular, for biometric applications [7-8].

In [7], a statistical framework modelling the co-update behaviour has been proposed in the biometric scenario. This model lies on hypotheses which fit the practical scenarios involving fingerprints and faces, providing a first analytical framework able to explain the relationship between the amount of unlabelled data collected and the template set size for a certain application. However, it can be shown that this model derives from a general framework which involves more than two matchers, independently on the peculiar application. The aim of the present work is to point out this fact by proposing the "co-update of multiple matchers". With this term, we mean a generic classification system based on template-matching [1]. Preliminary experiments are done on a case-study of a bi-modal biometric verification system. Results show the reliability of the framework by comparing the template set sizes and errors predicted by the model with the template set sizes and errors obtained on test data.

The paper is organized as follows. Section 2 describes the current version of the model. Section 3 reports the experimental results which validate the model, and also shows the difference with the previous one. Section 4 concludes the paper.

2 Co-updating of Multiple Matchers

Let us consider a pattern recognition problem characterized by c sets of features (here called "views"), and each view is conditionally independent of each other. The system is made up of c matchers, one matcher for each view. Thanks to the independence assumption, each recognizer is expected to assign correct "labels" to certain input data which are difficult for the other. In the case of template co-update, two matchers are used (bi-modal system) [7, 8]. In this Section, we show that the model proposed in [7] derives from a general framework involving more than two matchers.

The description of the generalized co-update algorithm is given in Figure 1. Let D_l be a set of labelled data. These data are the templates captured by human supervision, thus they are labelled with classes identifiers (e.g. users names in the case of personal verification). It is supposed that a batch D_u (usually, much larger than D_l) of data is acquired during system operation. Each element of D_u is a c-uple of samples, one sample for each view. Matchers can be combined in different kinds of ensembles (abstract-level, score-level [10]). Without loss of generality, we can neglect the particular kind of adopted fusion rule. By setting the size of the ensemble *Dim_ens*, $N\_ens = \binom{c}{Dim\_ens}$, different ensembles are possible. The main idea of co-updating of multiple matchers is that each ensemble can update the template set of remaining c-*Dim_ens* matchers. In this paper, we adopted the same terminology used in [7] for sake of uniformity with the state-of-the-art. Let us call "master" the ensemble that assume the supervisor's role, and "slave" the remaining matchers, whose template sets are augmented thanks to the master ensemble.

During the off-line co-update phase, each ensemble is applied to the batch D_u. The results of matching is a value which can be interpreted as a similarity value between the input sample and related template set. If this value exceeds a given "updating threshold", the match is verified, and 'slave' samples in the c-uple are added to the set

D_l. All matchers assume, alternatively, master and slave roles. The process is repeated a specified number of times.

It is worth highlight two particular cases. (1) *Dim_ens*=1: the ensemble is made up of only one matcher at time, the other ones are 'slaves'. (2) *Dim_ens*=*c*-1, the ensemble is made up of c-1 matchers at time, and the remaining one is 'slave'.

```
Define $c$ matchers, a matcher for each feature set x1, …, xc
Set the size of the ensemble $Dim\_ens$
Set the number of possible ensembles $N\_ens = \binom{c}{Dim\_ens}$

In the enrolment (supervised) session, collect a set $D_l$ of samples (in terms of c feature sets). A c-uple of samples $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_C\}$ – one sample for each matcher- is acquired.
Create templates using the set $D_l$
Loop for $H$ iterations:
    Collect a set $D_u$ without supervision. Each element of $D_u$ is a c-uple of samples $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_C\}$ from the same class
    For each ensemble $E_i$, $i \in \{1,2,...,N\_ens\}$
        Let the matchers used by $E_i$ be the 'master', while the others are the 'slaves'
        For each c-uple of elements in $D_u$
            If the ensemble $E_i$ (master) verifies the matching (i.e. match exceed a pre-defined updating threshold), the samples in the c-uple related to 'slave' matchers are added to the set $D_l$
Update templates using the augmented labelled set $D_l$
```

Fig. 1. Algorithm for Co-updating of multiple matchers

As for the majority of semi-supervised template update algorithms, the core of co-update is the insertion of new samples without external supervision. This might introduce misclassified samples in template set. For this reason, only "confident examples" are added by a very high co-update threshold value at which no false matches occur, namely, FAR = 0%. The role, master or slave, of the matchers is highlighted by superscripts M and S. For sake of clarity, we omit in the following terminology the explicit reference of each slave matcher.

Some assumptions are made in order to propose a mathematically tractable model. The first assumption regards the finite discrete space used to represent different possible values for each feature set (for example, a certain class can be represented by a finite number of possible directed acyclic graphs). Each class can produce a maximum of $N_{TOT}^{(i)}$ different observations for each of the *c* representations at hand. This assumption can be motivated easily noting that only a finite number of samples are collectable in a physical system. The main consequence of this assumption is that a new example, very similar to an existing one, can be considered already present in the template set.

The second assumption states that all possible observations are equally probable (e.g. this is usually stated in biometric applications), and the *c* representations are conditionally independent given the class [8,9]. In the co-update stage, a set D_u of $|D_u|=k$ *c*-uples of samples are presented to the system. We indicate with *k*' the number of samples of the master ensemble whose class has been verified (i.e. match exceeds the given updating threshold). The value *k*' can be easily computed by considering that each of *k* samples of the master ensemble can be drawn with replacement from a homogeneous population of $N_{TOT}^{(M)}$ elements. The probability that a sample will be verified is $p = 1 - FRR^{(M)}$, where $FRR^{(M)}$ is the False Rejection Rate for the master ensemble. The problem can be modeled using a binomial distribution. Therefore, on average:

$$k' = k\left(1 - FRR^{(M)}\right). \tag{1}$$

It is worth noting that the effectiveness of the master ensemble as supervisor towards the 'slave' matcher is related to its FRR value. If $FRR^{(M)}=0$ ($k'=k$) the master ensemble is equivalent to a "true" supervisor because it will verify the class of all 'genuine' samples in D_u. If $FRR^{(M)}>0$, then only $k'<k$ samples can be verified. Thus, only $k'<k$ samples will be added to the slave template set.

Only a finite number of samples are collectable in our physical system, so some samples appear indistinguishable. We are interested in samples that provide new information to the template set, so we can consider only distinguishable samples. Let *d* be the random variable representing the number of elements in *k'* that are different among themselves, and *dn* the random variable representing the number of elements in *k'* that are different and not present in the 'slave' template set. The pdf $p_d(d)$ can be modeled using a multinomial distribution:

$$p_d(d) = \binom{N_{TOT}^{(S)}}{d} \sum_{i=0}^{d} (-1)^i \binom{d}{i} \left(\frac{d-i}{N_{TOT}^{(S)}}\right)^{k'}. \tag{2}$$

The conditional pdf $p_{dn/d}(dn/d)$ – the probability to have *dn* different samples not present in the (S) template set if *d* samples are different among themselves – can be modeled using an hyper-geometric distribution:

$$p_{dn|d}(dn \mid d) = h\left(dn; N_{TOT}^{(S)}, (N_{TOT}^{(S)} - n), d\right). \tag{3}$$

where $N_{TOT}^{(S)}$ is the number of samples in the population, $(N_{TOT}^{(S)} - n)$ is the number of samples in the population that are classified as 'highly confident genuine samples' (i.e. samples actually belonging to the matched class), *d* is the number of items in the sample, *dn* is the number of items in the sample that are classified as 'highly confident genuine samples'. By using Eqs. (2) and (3), the pdf $p_{dn}(dn) = \sum_{d=1}^{k'} p_{dn/d}(dn/d)\, p_d(d)$ is derived. Here, we are interested to the expected value $E[dn]$. This represents the expected enlargement of the 'slave' template set due to the collection of *k'* elements. By assuming that for each co-update step the number $\Delta n^{(S)}$ of new and distinct samples added to the slave template set is about equal to its expected value $E[dn]$, since the correspondent variance var[*dn*] is very small:

$$\Delta n^{(S)} \approx E[dn] = \sum_{dn=0}^{k'} \{dn \cdot p_{dn}(dn)\} = \left(1 - \left(1 - \frac{1}{N_{TOT}^{(S)}} \right)^{k\left(1-FRR^{(M)}\right)} \right) \left(N_{TOT}^{(S)} - n^{(S)} \right) . \quad (4)$$

Let $n_0^{(S)}$ be the initial size of the (S) template set, and $n_{(i)}^{(S)}$ the size of the (S) template set at the *i*-th co-update step. The length of the (S) template set at the *i*-th step can be written as:

$$\begin{array}{l} n_{(0)}^{(S)} = n_0^{(S)} \\ n_{(i)}^{(S)} = n_{(i-1)}^{(S)} w + N_{TOT}^{(S)} (1-w) \end{array} ; \quad w = \left(1 - 1/N_{TOT}^{(S)}\right)^{k\left(1-FRR^{(M)}\right)} . \quad (5)$$

The weight *w* takes into account the effectiveness of the master template set as a supervisor. By recalling that in the ensemble of template matchers, as illustrated in Fig. 1, when a matcher plays the role of 'master' the other *c*-1 matchers play the role of 'slave', we can write the length of the *j*th (S) template set at the *i*-th step

$$\begin{array}{l} n_{(0)}^{(S\,j)} = n_0^{(S\,j)} \\ n_{(i)}^{(S\,j)} = n_{(i-1)}^{(S\,j)} w^{(j)} + N_{TOT}^{(S\,j)} \left(1 - w^{(j)}\right) \end{array} ; \quad w^{(j)} = \left(1 - 1/N_{TOT}^{(S\,j)}\right)^{k\left(1-FRR^{(M)}\right)} . \quad (6)$$

It is worth noting that while the effectiveness of the master template set as a supervisor depend on the FRR$^{(M)}$ value, the final growth of each (S*j*) template set depend on each (S*j*) biometric itself, via the parameter $N_{TOT}^{(S\,j)}$. The FRR$^{(M)}$ parameter is related to the behavior of the ensemble formed by the *c*-1 matchers.

The FRR$^{(M)}$ value determine the system's dynamics, so it is worth to relate its value to the length of the master template set. In order to model the above value each j-th sample can be characterized by a $m_{ij} \geq 0$ connection degree with other samples, and the *i*-th class can be characterized by the sequence $\{m_{i1}, m_{i2}, \ldots, m_{iN_{TOT}}\}$ of connection degrees. The "connection degree" is the number of examples that produce a similarity value over the updating threshold. There is a subset of samples for which this connection degree is zero. We refer to such samples with the term "isolated". We propose to characterize each client with two parameters: the fraction of samples that are isolated ($m_{ij} = 0$), namely, f_I, and the average connection degree $m_i = \langle \{m_{ij}\} \rangle$ computed only for the 'connected' samples. Therefore, "difficult" classes can be modeled by a low value of m_i and high value of f_I, whilst high value of m_i and low value of f_I, are adopted for "easy" classes. Parameters m_i and f_I depend on the dataset, on the employed matchers, and on the ensemble formed by the 'master' matchers.

Isolated samples contribute to the FRR with a constant value equal to 1, because they cannot "match" to other samples. The contribute to connected sample is computed as follows. Let $\hat{x}$ be a connected sample with connection degree m_i. Let *r* be the number of the connected samples of $\hat{x}$ in the master template set, $0 \leq r \leq m_i$. The identity of $\hat{x}$ will be correctly verified if at least one among its m_i connected samples is in the master template set, that is, if *r*>0. If *r*=0 - events with probability p(r=0) - even connected samples cannot "match" with samples in the master template set, so their contribute to the FRR value will be $p(r=0)$.

When $n<(N_{TOT}-m_i)$ the random variable r will be modeled by the hypergeometric distribution $h\left(r;N_{TOT}^{(M)},m_i,n^{(M)}\right)$, and the FRR value depend on both isolated and connected samples. When the template set reaches the size $n=N_{TOT}-m_i$ at least one of the m_i connected samples will be in the master template set, so $p(r=0)=0$ and the FRR value depend only on the fraction of isolated samples f_I. In other words, there is a "saturation" value for FRR, that is FRR($n>N_{TOT}-m_i$)$=f_I$.

$$FRR^{(M)}=\begin{cases} f_I\cdot 1+f_C\cdot 0 & if\, n^{(M)}>N_{TOT}^{(M)}-m_i \\ f_I\cdot 1+f_C\cdot h(r=0;N_{TOT}^{(M)},m_i,n^{(M)}) & if\, n^{(M)}\le N_{TOT}^{(M)}-m_i \end{cases}. \qquad (7)$$

Eq. (5-6) allows us to model the template set size increase during the co-update iterations, whilst eq. (7) models the FRR of the master ensembles. Notice that, in order to simplify the model, we adopted the average of m_{ij} for connected samples. This average value does not take into account isolated samples, but only connected samples. The above relationships can be used to predict the behaviour of co-update.

3 Experimental Results

In this Section, the case-study of a biometric verification system made up of a face and a fingerprint matchers is adopted for the experimental validation of the proposed model. In biometrics, a single sample is given by a couple of face and fingerprint images, from which two independent features sets are derived [2-3]. Images collected by the supervisor are the initial template set, which is called "gallery" in biometric applications.

Although the number of matchers is two, the model can be validated by hypothesizing that each matcher behaves as an ensemble of set of independent matchers applied to the same biometric. This agrees with observations in previous Section: the model proposed in [7] derives from the general framework proposed here. Therefore, the following conclusions can be applicable for the general case.

3.1 The Data Set

The data set adopted consists of 42 individuals composed of 20 face and fingerprint images for each individual, by keeping in mind the independence of face and fingerprint traits. The time span of both the collected data sets spans over one year. Forty-two frontal face images with 20 instances representing significant illuminations changes and variations in facial expressions per person were used from the Equinox corporation database [11]. The fingerprint data set has been collected by the authors using Biometrika Fx2000 optical sensor. The images are acquired with variations in pressure, moisture and time interval to represent large intra-class variations. The results are computed on five random coupling of face and fingerprint datasets and are averaged. Whilst minutiae are simply extracted from the fingerprint images after commonly used processing algorithms [2], PCA is computed on the whole data set and applied to face images in order to reduce the size of the overall feature space. 95% of energy is retained according to the current literature [3].

It is worth noting that fingerprint and face data sets are strongly different in terms of environmental conditions: the face one is notably "simpler" than the fingerprint one. We adopted so different data sets in order to show the effect of intra-class variations on the model prediction ability.

3.2 Experimental Results

First of all, we implemented a simple bi-modal identification system made up of a PCA-based face matcher and a fingerprint matcher using the "String" matching algorithm ("String" is based on minutiae points). We used the standard versions of these two recognition algorithms [2-3]. Then, we implemented the template co-update algorithm in Figure 1. Both the eigenspace of the PCA-based face matcher and the co-update threshold value at FAR = 0% are computed using the whole dataset, and has not been updated during the co-update process. This approach is common to some template update practices [5-9]. The update of face and fingerprint templates is performed simply by adding new examples to the user's template set.

In the experiments the initial template set has been set as follows. We selected, as the initial template in D_l for the i-th client, a sample whose connection degree is exactly m_i. In other word, the initial template is a connected sample 'near' to other m_i samples. In the proposed model the i-th client is characterized by (f_C, m_i) so this initial template is representative for the client. The rationale for this choice is to exclude outliers from the initial template set, likewise to what happens in real situations, where the initial template is chosen in completely supervised fashion.

We simulated the acquisition of a batch set D_u by generating several sets of k=10 couples – face and fingerprint – of 'genuine' examples, drawn with replacement from a homogeneous population of $N_{TOT}^{(face)} = N_{TOT}^{(fingerprint)} = 20$ samples. We are aware that adopted database size may not be very well appropriate for the task, but it respects, on average, the size adopted in other template update works reported in literature [5-9].

In order to set the correct parameters in the proposed model, for each client and for each biometric we computed a) the value f_c, that is, the fraction of samples that produce a 'score' over the threshold S^*, and b) the value m_i, that is, the integer nearest to the average connection degree of the connected samples. For each client, results are averaged on ten trials (experimental values) and predicted using parameters evaluated on the related training set. Table 1 gives a summary of these parameters. In particular, it is shown the ratio between the average fraction of connected samples and the related standard deviation (second column). The ratio between the average connection degree and the related standard deviation is also shown.

Table 1. Ratio between the average fraction of connected samples (f_c) and related standard deviation. The same ratio related to the connection degree m_s is also shown for face and fingerprint systems.

	E[f_c]/devst[f_c]	**E[m]/devst[m]**
Face	0,62	3,15
Fingerprint	0,71	1,33

It is worth noting that the difference among considered biometric systems is not related to the fraction of connected samples. In fact, the value of fingerprint and face matchers is about the same. This mean that, with respect to the average value, the related standard deviation is such that the differences two systems is not appreciable. This is confirmed by the average value of f_c, which is 0,62 for the face matcher and 0,71 for the fingerprint one.

Strong differences can be appreciated by observing the ratio between the average connection degree (*m*) and the related standard deviation. This indicated that face matcher exhibit a connection degree higher than that of fingerprint one, and a lower standard deviation. Therefore, face matcher can be considered (maybe due to Equinox data) "easier" than fingerprint one, thus it is expected a significant performance improvement for the face matcher, by using the co-update approach..

This hypothesis is confirmed by Figs. 2, which compares the values obtained on test data with the values predicted by the model. Reported trend is obtained by averaging values over all the clients. For each iteration of the algorithm depicted in Figs. 2, the FRR value evaluated at zeroFAR and the template set sizes are reported. The model is able to predict the experimental performance, with a negligible difference. For example, in a real scenario, due to the presence of isolated samples, the true value of FRR will be greater than 0 (FRR=1 if all the samples in template set are isolated), and this is respected by proposed model.

Fig. 2 shows the average values over all the client. Discrepancy between predicted and experimental values can be justified recalling that the proposed model predicts the system's performance for each client separately. Moreover, the whole connection graph that characterizes each client is modelled using only two parameters (m_i and f_I). The saturation of theoretical and experimental curves is driven by eqs. (6-7). In particular, FRR of the master matcher is responsible of the weight *w*, which tends to decrease as the template set size of both matchers increases. Thus, the size of galleries

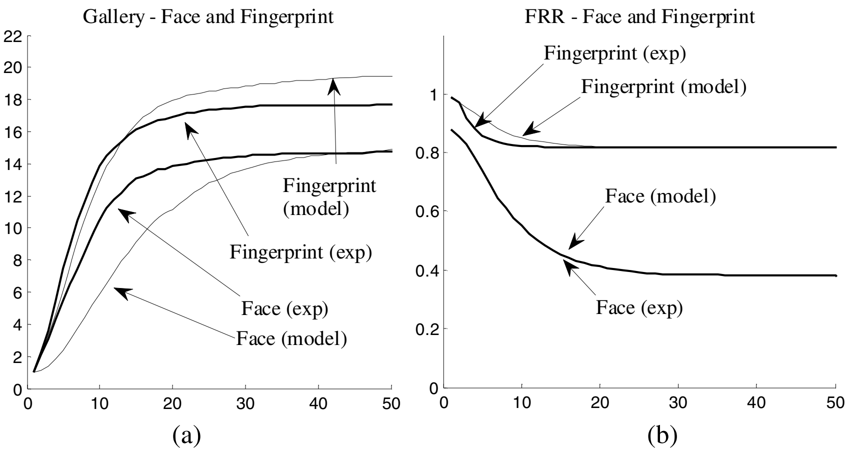


Fig. 2. Experimental trend and predicted values of template set sizes (a) and FRR (b) for fingerprint and face biometrics. Reported trend is obtained averaging values over all the clients.

must converge to N_{TOT} if m_i is not zero. At the end of co-update process, FRR(n=N_{TOT})=f_I. It is worth noting that the correct prediction of FRR values is matter of primary importance both for the validation of the model and for the designers, that need relevant information about the performance of the system. With regard to this issue, the FRR predicted by the proposed framework does not depend on the number of matchers, but only the template set size and related parameters (connection degree and fraction of connected samples).

An important problem to solve is how to compute the above parameters. At the state of our knowledge, this can be only done by collecting an additional set of examples. But this implies to verify that these parameters are not dependent on intra-class variations, but to intrinsic characteristic of each client. This will be matter of a future work.

4 Conclusions

In this paper we showed that the theoretical framework of template co-update can be derived from a general model proposed here. Preliminary experiments on a biometrics-based case-study have shown the effectiveness of the model. A further validation step is necessary by increasing the number of biometrics.

Other issues still remain, but we believe that the template co-update algorithm is well explained by proposed model. Therefore, it is suitable for designing classification systems, based on template matching, which can improve with use.

References

1. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley-Interscience, Hoboken (2000)
2. Maltoni, D., Maio, D., Jain, A.K., Prabhakar, S.: Handbook of fingerprint recognition. Springer, Heidelberg (2003)
3. Li, S.Z., Jain, A.K. (eds.): Handbook of face recognition. Springer, Heidelberg (2005)
4. Ross, A., Nandakumar, K., Jain, A.K.: Handbook of Multibiometrics. Springer, Heidelberg (2006)
5. Zhu, X.: Semi-supervised learning literature survey, Technical report, Computer Sciences TR 1530, Univ. Wisconsis, Madison, USA (2006)
6. Roli, F., Marcialis, G.L.: Semi-supervised PCA-based face recognition using self-training. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR 2006 and SPR 2006. LNCS, vol. 4109, pp. 560–568. Springer, Heidelberg (2006)
7. Didaci, L., Marcialis, G.L., Roli, F.: Modelling FRR of Biometric Verification Systems using the Template Co-update Algorithm. In: 3rd IAPR/IEEE Int. Conference on Biometrics ICB 2009, Alghero (Italy), June, 2-5. LNCS, Springer, Heidelberg (2009) (in press)
8. Roli, F., Didaci, L., Marcialis, G.L.: Adaptive biometric systems that can improve with use. In: Ratha, N., Govindaraju, V. (eds.) Advances in Biometrics: Sensors, Systems and Algorithms, pp. 447–471. Springer, Heidelberg (2008)
9. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proc. of the Workshop on Computational Learning Theory, pp. 92–100 (1998)
10. Kittler, J., Hatef, M., Duin, R., Matas, J.: On Combining Classifiers. IEEE Trans. On PAMI 20(3), 226–239 (1998)
11. `http://www.equinoxsensors.com/products/HID.html`

Handling Multimodal Information Fusion with Missing Observations Using the Neutral Point Substitution Method

David Windridge[1], Norman Poh[1], Vadim Mottl[2], Alexander Tatarchuk[2], and Andrey Eliseyev[2]

[1] CVSSP, University of Surrey, The Stag Hill, Guildford, GU2 7XH, UK
[2] Computing Center of the Russian Academy of Sciences, Vavilov St. 40, Moscow, 119991, Russia

Abstract. We have previously introduced, in purely theoretical terms, the notion of neutral point substitution for missing kernel data in multimodal problems. In particular, it was demonstrated that when modalities are maximally disjoint, the method is precisely equivalent to the Sum rule decision scheme. As well as forging an intriguing analogy between multikernel and decision-combination methods, this finding means that the neutral-point method should exhibit a degree of resilience to class misattribution within the individual classifiers through the relative cancelling of combined estimation errors (if sufficiently decorrelated).

However, the case of completely disjoint modalities is unrepresentative of the general missing data problem. We here set out to experimentally test the notion of neutral point substitution in a realistic experimental scenario with partially-disjoint data to establish the practical application of the method. The tested data consists in multimodal Biometric measurements of individuals in which the missing-modality problem is endemic. We hence test a SVM classifier under both the modal decision fusion and neutral point-substitution paradigms, and find that, while error cancellation is indeed apparent, the genuinely multimodal approach enabled by the neutral-point method is superior by a significant factor.

1 Introduction

In a paper given at the last MCS meeting [9], we set out a strategy for addressing the problem of missing modalities in multimodal kernel data. The problem of missing features is well-known in general pattern recognition, but can be addressed (aside from simply omitting the missing-data samples) using methods such as mean substitution [1], at the simplest level, or else via more complex methods (eg [3]) that take into account specifics of the distribution statistics and morphology.

However, in multimodal kernel decision problems the issue of missing features becomes acute (multimodal kernel decision problems are those in which feature maps $\hat{\phi}$ giving $\mathcal{R}^N$ outputs for detected objects ω are associated either with particular sensor spaces; $\hat{\phi^m}(S_m(\omega)) \rightarrow \mathcal{R}^{N_m}$, or else with particular kernel measures $K_m(\hat{\phi}^m(\cdot), \hat{\phi}^m(\cdot)) \rightarrow \mathcal{R}$ on some, possibly even *common*, sensor-output space S)[1]. The difficulties arise because we cannot, in general, assume that the Kernel matrix $\mathbf{K_n} = K_n(\hat{\phi}^n(S(\omega_i)), \hat{\phi}^n(S(\omega_j)))$ defined on a per-mode basis (ie applicable only to mode n) will give rise to the *same* Mercer embedding space, $\hat{\psi}^n(S) = (\psi_1^n(S), \psi_2^n(S), \psi_3^n(S), \ldots)'$ when the set from which i and j are drawn has differing cardinalities, r, due to the missing data[2]. (The functions $\psi_i^n(S)$ being Eigenfunctions of the Kernel matrix $\mathbf{K_n}$; ie such that $\hat{\phi}(S(\omega_i)) = \lambda^{\frac{1}{2}} u_i$, where $\mathbf{K_n} = \mathbf{U \Lambda U'}$ and $\mathbf{U} = (u_1, u_2, u_3, \ldots u_r)$, with $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, \ldots \lambda_n)$ the eigenvalue matrix, and $u_i = \psi_i(S(\omega_i)))$.

We cannot therefore simply assume that modes can be combined into a composite (Mercer) pattern space in which to perform classification (as in standard pattern recognition)[3]. Furthermore, even if this composition of spaces can be achieved, there are no kernels defined *a priori* within it since there are no *inter*-modal kernels defined at the outset, as would be required for pattern recognition based on Euclidean (l^2-norm), or quasi-Euclidean (l^n-norm) assumptions. Consequently, there is an ambiguity as to how the problem should be approached.

We therefore, rather, approach the problem from the opposite direction, firstly defining a composite kernel capable of accommodating missing Kernel values, and only secondarily considering the nature of the space in which this composite kernel is embedded, consistent with the ideals of kernel-based approaches generally.

The paper is therefore structured as follows: in the following section we recap the neutral-point approach to missing value substitution, and indicate its relation to the Sum-Rule decision scheme. In section 3 we detail the application of the method to multi-modal Biometric data (data in which the missing value problem arises naturally). Section 4 discusses experimental outcomes and makes concluding remarks.

---

[1] In which case the problem is similar to that of multikernel learning [6]. However, we here assume that that the association of kernels with output sensors with is in some way intrinsic to the experimental setup (e.g. the association of genetic-distance with gene-data), rather than being determined after the fact via an optimisation process as is sometimes the case in Multikernel learning).

[2] We could, of course, simply overlook this issue, and combine outputs in the sensors' tensor space; however this is to make the notion of using *mode-specific* Kernels redundant, in particular, losing the inherent advantages of the problem-relevant, minimalist and linearised embedding spaces so constructed. Furthermore, in certain domains (eg step-wise gene-distance measurements [8]), where there is no absolute underlying sensor space, it is only possible to work with kernel embedding spaces.

[3] This is not necessary problematic for other Kernel applications. For instance, a method for approaching the missing data issue at the objective function level in Kernel PCA is given in [7]; it utilises cross entropy with respect to a Gaussian distribution as the cost function to be minimised with respect to the missing values. However, this makes implicit assumptions about the data (namely that it can be modelled as a Gaussian with the given kernel), and takes a significant amount of time to compute.

2 The Neutral Point Method

We here recap the essentials of the neutral point method; for more detailed information refer to [9]. For clarity in this section, we assume an underlying unidimensional sensor space within each mode, and omit explicit consideration of the sensor-space/feature-map relation $\phi(S(\omega))$ as it does not effect findings:

We thus consider a set of Kernel measures, K_i in relation to which sensor outputs can be defined for each entity ω (ie where x maps objects ω into a common real valued space):

$$\mathcal{X}_i = \{x(\omega), \omega \in \Omega\} \tag{1}$$

Any kernel $K_i(x_i', x_i'')$ embeds the scale of the respective sensor $\mathcal{X}_i$ (equipped with with inner product) into a hypothetical linear space (the embedding space), $\hat{\mathcal{X}}_i \supseteq \mathcal{X}_i$, in which the null element and linear operations are defined.

For a single modality, the training set:

$$\Omega_i^\star = \{\omega_j, j = 1, \ldots, N_i\} \tag{2}$$

is *completely* defined by kernel matrix and class indices y ($y = \pm 1$):

$$\Omega_i^\star => \{\mathbf{K}_i = \lfloor K_i(x_i(\omega_j), x_i(\omega_l)), \omega_j, \omega_l \in \Omega_i^\star \rfloor, y(\omega_j), \omega_j \in \Omega_i^\star\} \tag{3}$$

Support Vector Machines (SVMs) are the most common Kernel-based approach approach to 2-class pattern recognition, the problem being to find maximal margin discriminant hyperplane in space $\boldsymbol{\mathcal{X}}_i$:

$$\boldsymbol{y}_i(x_i(\omega)) = K_i(\theta_i, x_i(\omega)) + b_i \overset{>}{<} 0 \tag{4}$$

(which generally has a much more complex decision boundary in $\mathcal{X}_i$).

This leads to the standard SVM Training Criterion:

$$K_i(\theta_i, \theta_i) + C \sum_{\omega_j \in \Omega_i^\star} \delta_j \to \min(\theta_i \in \boldsymbol{\mathcal{X}}_i, b \in \mathcal{R}, \delta_j \in \mathcal{R}) \tag{5}$$

Subject to:

$$y_i \lfloor K_i(\theta_i, x_i(\omega_l)) + b \rfloor \geq 1 - \delta_j, \delta_j \geq 0 \tag{6}$$

The (Wolfe) dual form of the criterion is a quadratic programming problem with respect to the Lagrangian multipliers, λ:

$$\sum_{\omega_j \in \Omega_i^\star} \lambda_{i,j} - (1/2) \sum_{\omega_j \in \Omega_i^\star} \sum_{\omega_l \in \Omega_i^\star} \lfloor y_j y_l K_i(x_i(\omega_j), x_i(\omega_l)) \rfloor \lambda_{i,j} \lambda_{i,l} \to \max \tag{7}$$

Subject to:

$$\sum_{\omega_j \in \Omega_i^\star} y_j \lambda_{i,j} = 0, 0 \leq \lambda_{i,j} \leq C/2, \omega_j \in \Omega_i^\star \tag{8}$$

This gives rise to the usual decision rule defined by the support objects $\hat{\Omega}_i \in \Omega_i^\star$ as the remaining Lagrange multipliers tend to zero $\lambda_{i,j} \to 0$ (leaving $\hat{\lambda}_{i,j} > 0$):

$$\hat{f}(x_i(\omega)) = \sum_{j:\omega_j \in \Omega_i^\star} y_j \hat{\lambda}_{i,j} K_i(x_i(\omega_j), x_i(\omega_l)) + \hat{b}_i \gtrless 0 \tag{9}$$

with:

$$\hat{b}_i = -\left(\sum_{j:\omega_j \in \Omega_i^\star} \hat{\lambda}_{i,j} \sum_{l:\omega_l \in \Omega_i^\star} y(\omega_l) \hat{\lambda}_{i,l} K_i(x_i(\omega_j), x_i(\omega_l)) / \sum_{j:\omega_j \in \Omega_i^\star} \hat{\lambda}_{i,j} \right) \tag{10}$$

However, there exits a continuum of points for each i for which no decision is given:

$$\hat{x}_{\phi,i} \in \boldsymbol{\mathcal{X}}_{\phi,i}, \boldsymbol{\mathcal{X}}_{\phi,i} = \{x_i \in \boldsymbol{\mathcal{X}}_i : K_i(\hat{\theta}_i, x_i) + \hat{b}_i = 0\}, \hat{b}_i = -K_i(\hat{\theta}_i, x_{\phi,i}) \tag{11}$$

These are the neutral points. In the following, we do not, at any stage, need to explicitly calculate them. In particular, where an individual neutral point is used in calculation, we shall find that it is only required that the neutral point be one drawn from the total set of of neutral points, without having the requirement of specifying *which* neutral point it is. In other words the designator of an individual neutral point behaves like a 'particularity' operator and not an indexical operator.

To proceed further, we now need to explicitly consider the multikernel decision problem. Substituting the most straightforward multi-modal Kernel, the linear kernel where $K(x', x'') = \sum_{i=1}^{n} K_i(x'_i, x''_i)$ into the (non-dual) SVM decision problem, we find that the training criterion becomes:

$$K_i(\theta_i, \theta_i) + C \sum_{\omega_j \in \Omega_i^\star} \delta_j \to \min(\theta_i \in \boldsymbol{\mathcal{X}}_i, b \in \mathcal{R}, \delta_j \in \mathcal{R}) \tag{12}$$

Subject to:

$$\lfloor y_j (K_i(\theta_i, x_i(\omega_j)) + \sum_{l=1, l\neq i}^{n} K_l(\theta_l, x_l(\omega_j)) + b) \geq 1 - \delta_j, \delta_j \geq 0, \omega_j \in \Omega_i^\star \rfloor, i = 1, \ldots, n \tag{13}$$

However, the question arises as to the existence of the terms $K_l(\theta_l, x_l(\omega_j))$, when $l \neq i$; that is, where an object designated within one mode's kernel embedding space also exists within another mode's kernel space. If, for instance, multi-modal training sets are partially disjoint (e.g. when training sets have missing feature values) then the multi-mode kernel problem is not soluble in itself. If multi-modal training sets are completely disjoint (for instance, when the training sets within each mode are proprietary) then the multi-modal kernel problem is maximally intractable.

However, because of the presence of the individual modes' decision problems in the above constraint optimisation problem, we can apply the neutral point substitution as constituting the least biasing value substitution *given the decision problem in question.* Thus, rather than proposing a missing data approach that makes strong assumptions about the form of the data (perhaps that it is Gaussian in nature), or else takes only very partial consideration of the nature of the data (as in mean-substitution), we propose to adopt a missing-data approach that is *relevant to the classification problem in hand.*

Hence, we the replace 'missing' sensor values $x_l(\omega_j), l \neq i$, by unbiased neutral points: $\hat{x}_{\phi,i} \in \hat{\mathcal{X}}_{\phi,i}$.

It was shown [9] that, in the case of completely disjoint modalities, the solution to the above equation with appropriate neutral point substitutions (such that $K_l(\theta_l, x_l(\omega_j)) + b = 0$) becomes linearly separable in b, and defaults to the sum rule decision scheme for the individual modes' SVMs:

$$\hat{f}(x_i(\omega), i =, \ldots, n) = \sum_{1=1}^{n} \lfloor K_i(\hat{\theta}_i, x_i(\omega_l)) + \hat{b}_i \rfloor \gtrless 0 \tag{14}$$

This is a very reassuring result, in that it shows that our choice of unbiased substitution for missing data naturally corresponds to the only alternative way of dealing with the completely disjoint data problem (ie treating it as a case of decision fusion). Further, it indicates that neutral point substitution readily permits room for the error decorrelation effect to take place (which can be important if the composite Kernel increases the dimensionality of the embedding space to the point at which the 'curse of dimensionality' becomes apparent). What is not immediately clear, however, is the extent to which this effect is advantageous for partially disjoint data, where the composite Mercer space is not so straightforwardly decomposable into its marginal components. In this case, we can regard the the completed data (ie the data without missing components) as 'weighting' the summed marginal decisions on the basis of the intra-modal correlations to an extent that is governed by their proportion of the total data. The exact degree to which this occurs will be data and kernel dependant. We would therefore like to quantify this result for a typical data set.

We hence now turn to an empirical exploration of the neutral point method in a realistic scenario, in which the modal data is only very partially disjoint; that is, where the multimodal data is largely complete, apart from a few missing values (eg, of the sort that occur in the field of census data returns).

3 Experimental Findings

3.1 Database, Reference Systems and Experimental Protocols

The data used in our evaluation scheme is taken from the Biosecure database. *Biosecure*[4] is a European project whose aim is to integrate multi-disciplinary research efforts in biometric-based identity authentication. Application examples are a building access system using a desktop-based or a mobile-based platform,

[4] http://www.biosecure.info/

Table 1. A list of channels of data for each biometric modality captured using a given device

(a) Channels of data

Label	template ID {n}	Modality	Sensor	Remarks
fa	1	Still Face	web cam	Frontal face images (low resolution)
ft	1–6	Fingerprint	Thermal	1/4 is right/left thumb; 2/5 is right/left index; 3/6 is right/left middle finger
ir	1–2	Iris image	LG	1 is left eye; 2 is right eye

(b) Reference systems

Modality	Reference systems
Still Face	Omniperception's Affinity SDK face detector; LDA-based face verifier
Fingerprint	NIST Fingerprint system
Iris	A variant of Libor Masek's iris system

(c) Protocols

Data sets		No. of match scores per person	
		dev (51 persons)	eva (156 persons)
S1	Gen	1	1
	Imp	103×4	51×4
S2	Gen	2	2
	Imp	103×4	126×4

as well as applications over the Internet such as tele-working and Web or remote-banking services. As far as the data collection is concerned, three scenarios have been identified, each simulating the use of biometrics in remote-access authentication via the Internet (termed the "Internet" scenario), physical access control (the "desktop" scenario), and authentication via mobile devices (the "mobile" scenario). A report on the complete Biosecure database is being drafted.

For the purpose of our experiments, we used the subset of desktop scenario, which further contains a subset of still face, 6 fingers and iris modalities, denoted by fa1, ft1–6 and ir1, respectively. These 8 channels of data, as well as the reference system, and the experimental protocols are summarized in Table 1.

Note that for the purpose of performance assessment, the main objective of this paper, the data set and experimental protocols are not the primary concern; any database could have been used. The only requirement is that a wide variety of biometric modalities are used in order to illustrate the generality of our approach.

It is important to note that there are two score data sets: development and the evaluation sets (see Table 1(c)). In this table, S1 means the session 1 data whereas S2 means the session 2 data. The data in S1 consists of two samples collected within the same session. They are collected to facilitate the development of a baseline system. It is known that intra-session performance is biased [4]. For this reason, we shall use the S2 data for our evaluation. A plot of EER for the 8 channels of data is shown in Figure 1. The iris baseline system used here is far from the performance claimed by Daugman's implementation [2]. We verified that this is due to bad iris segmentation and a suboptimal threshold for distinguishing eyelashes from iris (being baselines, no effort was made to optimize performance; the only requirement is that all systems output match scores).

Two factors can result in missing modalities. First, during the data collection process, some volunteers did not complete a whole session. Second, some acquired

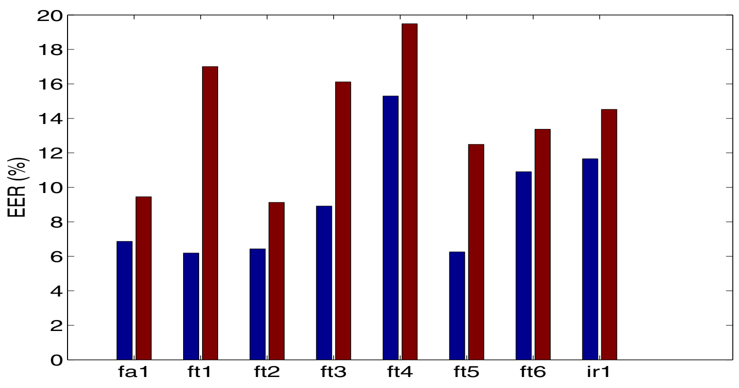


Fig. 1. The error of the development set (blue) versus that of evaluation set (red) of the 8 systems used in the cost-sensitive evaluation of the Biosecure data set

Table 2. Correlation matrix of genuine scores. fa1=face, ft1–6: fingerprints, ir1=iris match scores.

fa1	ft1	ft2	ft3	ft4	ft5	ft6	ir1
1.00	0.15	-0.03	-0.07	-0.13	0.12	-0.08	-0.07
0.15	1.00	0.33	0.40	0.39	0.56	0.44	0.08
-0.03	0.33	1.00	0.60	0.41	0.58	0.64	0.02
-0.07	0.40	0.60	1.00	0.51	0.62	0.66	0.02
-0.13	0.39	0.41	0.51	1.00	0.51	0.56	-0.04
0.12	0.56	0.58	0.62	0.51	1.00	0.73	-0.12
-0.08	0.44	0.64	0.66	0.56	0.73	1.00	-0.11
-0.07	0.08	0.02	0.02	-0.04	-0.12	-0.11	1.00

biometric samples are so low in quality that they cannot be processed by our feature extraction algorithm, or the resultant extracted features could not be used for matching. Being well controlled, the development set contains almost complete observations; however a fraction of samples in the evaluation set (8348 out of 76920) contain some missing modalities.

3.2 Correlation Analysis of the Match Scores

We may summarise the data as paired biometric systems delivering impostor match scores[5] (the corresponding genuine user match scores are similar and, hence not considered here).

In particular, it is useful to summarize the two class-conditional covariance matrices by their correlation matrices since correlation is invariant to variable scaling and is bounded in $[-1, 1]$, with 1 (resp. -1) being perfect positive (resp.

[5] Match scores used in the experiments are available for download at: http://personal.ee.surrey.ac.uk/Personal/Norman.Poh/web/fusionq

Table 3. Impostor scores

1.00	-0.02	-0.02	-0.05	-0.01	-0.03	-0.05	0.10
-0.02	1.00	0.06	0.08	0.09	0.04	0.07	-0.03
-0.02	0.06	1.00	0.12	0.05	0.14	0.10	-0.03
-0.05	0.08	0.12	1.00	0.08	0.10	0.15	-0.05
-0.01	0.09	0.05	0.08	1.00	0.05	0.07	-0.02
-0.03	0.04	0.14	0.10	0.05	1.00	0.14	-0.03
-0.05	0.07	0.10	0.15	0.07	0.14	1.00	0.02
0.10	-0.03	-0.03	-0.05	-0.02	-0.03	0.02	1.00

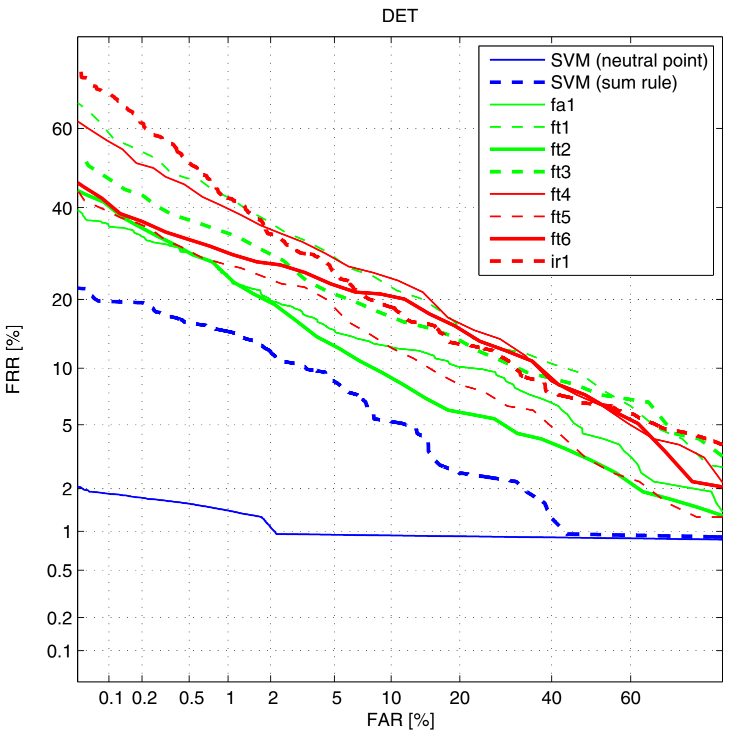


Fig. 2. Performance of the baseline expert systems and that of fusion with SVM using the sum rule and the neutral point substitution method

negative) correlation. The correlation matrix of the impostor and client match scores calculated on the development set are shown in Table 2 and Table 3.

There are three points to note. First, the impostor match scores have generally correlation entries close to zero. Second, the correlation among all the six fingers (columns 2 to 7, resp. rows 2 to 7) are *all* positive, albeit having small values. Third, the correlation among the genuine match scores of all the six fingers (columns 2 to 7, resp. rows 2 to 7) have relatively high values (from 0.3 to 0.6). According to [5], this indicates that combining two fingerprint systems may not be as effective as combining two different biometric traits, e.g., a fingerprint and a face biometric. The problem is therefore implicitly *multi*-modal, and can be kernelised in terms of SVM recognition within the individual modes.

3.3 Results

Using the neutral point substitution method outlined in section 2, we therefore specified an experimental scenario in which the SVM classifier acts both individually upon the modalities of the Biosecure database, and collectively via sum rule decision fusion and composite kernelization. Composite kernelisation is carried-out via the linear kernel $K(x', x'') = \sum_{i=1}^{n} K_i(x'_i, x''_i)$ with neutral point substitution undertaken for the missing values. An inner product kernel is chosen for transparency within the individual modalities.

The results of these tests are given as superimposed ROC curves in Figure 2.

4 Discussion and Conclusions

It was demonstrated theoretically that the neutral point method is an appropriate strategy for treating missing values in multi-kernel problems with the potential to retain the error-decorrelation advantages of the sum-rule decision scheme in typical test scenarios with partial missing data. Experiments were consequently conducted on multimodal biometric data from the Biosecure database, in which both multi-kernelisation and the missing data problem arose naturally, in order to complement the theoretical analysis derived for the asymptotic scenario of complete data-disjunction.

Results (Fig. 2) demonstrate that the sum rule decision scheme is indeed superior to any individual modal decision rule on the tested data, but that very significantly greater advantage arose from the composition of Kernels (which would, in itself, be impossible without missing value substitution). We hypothesise that this result will be typical for naturally-arising multi-kernel, missing-data problem (data in which missing values are relatively rare). The neutral point method is thus an appropriate 'first-resort' strategy to consider in these cases, as opposed to modal fusion; particularly as the latter is implicit in the former.

Because of the nature of the derivation of the neutral point method, there is no explicit requirement for actual value substitution, and the method gives rise to minimal changes to the cost function of linearised kernel composition. Furthermore, the method differs from previous approaches in that the missing values are related to the decision problem rather than to the data distribution. In this way it is consistent with the broad philosophy of maxim margin SVM-based approaches. We thus conclude that the neutral point method can be characterised as an empirically safe and theoretically-unbiased approach to missing data substitution.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^o 215078.

This work was supported partially by the advanced researcher fellowship PA0022_121477 of the Swiss National Science Foundation, and by the EU-funded Mobio project grant IST-214324 (www.mobioproject.org).

References

1. Cuesta, J.M.L., de Cordoba Herralde, R., D'Haro Enriquez, L.F.: Applying feature reduction analysis to a pprlm-multiple gaussian language identification system. In: Articulo en actas de las V Jornadas de Tecnologia, pp. 29–32 (2008)
2. Daugman, J.: How Iris Recognition Works, ch. 6. Kluwer Publishers, Dordrecht (1999)
3. Lin, T.I., Lee, J.C., Ho, H.J.: On fast supervised learning for normal mixture models with missing information. Pattern Recognition 39(6), 1177–1187 (2006)
4. Martin, A., Przybocki, M., Campbell, J.P.: The NIST Speaker Recognition Evaluation Program, ch. 8. Springer, Heidelberg (2005)
5. Poh, N., Bengio, S.: How Do Correlation and Variance of Base Classifiers Affect Fusion in Biometric Authentication Tasks? IEEE Trans. Signal Processing 53(11), 4384–4396 (2005)
6. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: Simplemkl. Journal of Machine Learning Research (2008)
7. Sanguinetti, G., Lawrence, N.D.: Missing data in kernel pca (2006), http://www.dcs.shef.ac.uk/intranet/research/resmes/CS0608.pdf
8. Walsh, B.: Estimating the time to the mrca for the y chromosome or mtdna for a pair of individuals. Genetics 158, 897–912 (2001)
9. Windridge, D., Mottl, V., Tatarchuk, A., Eliseyev, A.: The neutral point method for kernel-based combination of disjoint training data in multi-modal pattern recognition. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 13–21. Springer, Heidelberg (2007)

Influence of Hyperparameters on Random Forest Accuracy

Simon Bernard, Laurent Heutte, and Sébastien Adam

Université de Rouen, LITIS EA 4108
BP 12 - 76801 Saint-Etienne du Rouvray, France
{simon.bernard,laurent.heutte,sebastien.adam}@univ-rouen.fr

Abstract. In this paper we present our work on the Random Forest (RF) family of classification methods. Our goal is to go one step further in the understanding of RF mechanisms by studying the parametrization of the reference algorithm Forest-RI. In this algorithm, a randomization principle is used during the tree induction process, that randomly selects K features at each node, among which the best split is chosen. The strength of randomization in the tree induction is thus led by the hyperparameter K which plays an important role for building accurate RF classifiers. We have decided to focus our experimental study on this hyperparameter and on its influence on classification accuracy. For that purpose, we have evaluated the Forest-RI algorithm on several machine learning problems and with different settings of K in order to understand the way it acts on RF performance. We show that default values of K traditionally used in the literature are globally near-optimal, except for some cases for which they are all significatively sub-optimal. Thus additional experiments have been led on those datasets, that highlight the crucial role played by feature relevancy in finding the optimal setting of K.

Keywords: Supervised Learning, Ensemble Method, Random Forests, Decision Trees.

1 Introduction

Random Forest is a family of classifier ensemble methods that use randomization to produce a diverse pool of individual classifiers, as for Bagging [1] or Random Subspaces methods [2]. It can be defined as a generic principle of classifier ensemble that uses L tree-structured base classifiers $\{h(x, \Theta_k),\ k = 1, ...L\}$ where $\{\Theta_k\}$ is a family of independent identically distributed random vectors, and x is an input data. The particularity of this kind of ensemble is that each decision tree is built from a random vector of parameters. A Random Forest can be built for example by randomly sampling a feature subset for each decision tree (as in Random Subspaces), and/or by randomly sampling a training data subset for each decision tree (as in Bagging). Since they have been introduced in 2001, RFs have been studied in many ways, theoretically as well as experimentally [3,4,5,6,7,8]. In most of those works, it has been shown that RFs are particularly competitive with one of the most efficient learning principles, *i.e.* boosting [4,8]. However, the mechanisms that explain the good performance of RFs are not clearly identified and one has to admit that it is still a complex task for the practitioner to take full benefits of the potential of those methods. For example considering the reference RF method called Forest-RI, introduced by Breiman in [4] (see section 2), an important hyperparameter has been identified : the number K of features randomly selected at each node during the tree induction process. Yet, in those research works that have experimented this method, the value of K is arbitrarily or empirically set, and sometimes without any theoretical nor experimental justification.

In this paper we propose to go one step further in the understanding of RF mechanisms, by studying the parametrization of the reference algorithm Forest-RI. We propose to study the influence of the hyperparameter K on the classification accuracy, in order to empirically distinguish parametrization rules or tendencies, according to some characteristics of the classification problem. The final goal is to give elements to the practitioner in order to help him building RFs that accurately suit to the specificities of a classification problem. For that purpose we have experimented on several machine learning datasets the algorithm Forest-RI with different settings of K in order to study their influence on accuracy. We show that default values of K traditionally used in the literature are globaly near-optimal, except for some cases for which they are all significatively sub-optimal. We have then studied the relation between the nature of the feature space and RF performance, by studying feature relevancy. We highlight the crucial role played by feature relevancy in finding the optimal setting of K.

The paper is thus organized as follows: we recall in section 2 the Forest-RI principles and related works on its experimentation; in section 3, we describe our experimental protocol, the datasets used, and the results obtained with different settings of the hyperparameter K. We finally draw some conclusions and future works in the last section.

2 The Forest-RI Algorithm

One can see Random Forests as a family of methods, made of different decision tree ensemble induction algorithms, such as the Breiman Forest-RI method often cited as the reference algorithm in the literature [4]. In this algorithm the Bagging principle is used with another randomization technique called Random Feature Selection. The training step consists in building an ensemble of decision trees, each one trained from a bootstrap sample of the original training set — *i.e.* applying the Bagging principle — and with a decision tree induction method called Random Tree. This induction algorithm, usually based on the CART algorithm [9], modifies the splitting procedure for each node, in such a way that the selection of the feature used for the splitting criterion is partially randomized. That is to say, for each node, a feature subset is randomly drawn, from which the best splitting criterion is then selected as in traditionnal tree induction algorithms. To sum up, in the Forest-RI method, a decision tree is grown by using the following process :

- Let N be the size of the original training set. N instances are randomly drawn with replacement, to form the bootstrap sample, which is then used to build the tree.
- Let M be the dimensionality of the original feature space, and K a preliminary fixed hyperparameter so that $K \in [1, M]$. For each node of the tree, a subset of

K features is randomly drawn without replacement, among which the best split is then selected.

- The tree is thus built to reach its maximum size. No pruning is performed.

In this process the tree induction is directed by a single hyperparameter, *i.e.* the number K of randomly selected features. This number allows to introduce more or less randomization in the induction. Whereas this hyperparameter seems to be critical to induce accurate RF [3], no research work has been specifically devoted to the study of its setting and its real influence on performance, and only a few have empirically dealt with this issue.

In [6] for example, Geurts et al. have proposed a new method of RF induction, called Extras-Trees for Extremely Randomized Tree Ensemble, that modifies the Forest-RI algorithm to accentuate the randomization. Here the Random Feature Selection is still used but modified so that the best splitting criterion selection is one step further randomized. The authors have designed their experimental protocol to study the influence of K on performance. Even if this method is partly different from the Forest-RI algorithm, this work allows to draw some intuitions on the Random Forest behavior according to K. It highlights for example that its default setting $K = \sqrt{M}$, where M stands for the size of the original feature set, is most of times closed to the optimal setting, at least for the Extras-Trees method and on several representative datasets.

When introducing RF formalism, Breiman studied performance according to K [4]. In these experiments, a large number of RF has been grown on three datasets, for which the test set error rate has been monitored. Actually only one of those three experiments was really concerned by the Forest-RI algorithm, since the two others have been run with an induction algorithm that uses feature combinations as splitting criterions, instead of single features. Hence, even if some tendencies can be intuitively guessed, those experiments do not allow to conclude on RF behavior according to the setting of K. We also noticed that in his Forest-RI experiments, Breiman decided to use two values of $K : 1$ and $log_2M + 1$. While the first value is intuitively interesting since it corresponds to a decision tree induction that selects in a fully random manner the splitting criterion among features for each node, the second one seems to be more arbitrary or at least is not justified.

In [3], a serie of tests with Forest-RI has been led on the well-known MNIST handwritten digit recognition dataset [10]. An interval of values of K has been found for which best accuracies have been reached; this interval does not contain neither $K = 1$ nor $K = M$ but contains the two values $K = \sqrt{M}$ and $K = log_2(M) + 1$. However we think that their primary conclusions need to be confirmed with a more rigourous experimental protocol and with several different machine learning datasets.

Finally, implementation and experimentation of the Forest-RI algorithm require to fix the value of the hyperparameter K but there actually does not exist any theoretical rule that can be used to fix it. As mentioned previously, only arbitrary default values are proposed in the literature and nothing guarantees that these values are close to the optimal setting. Thus one of the goal of the work presented in this paper is to bring elements of RF mechanism understanding by focusing on the hyperparameter K and on the way it acts on RF accuracy. For that purpose we have led a set of experiments that are described in the following section. Notice that in the rest of this paper, the term Random Forest (RF) will always stand for a forest induced with the Forest-RI algorithm.

3 Investigating the Influence of Hyperparameter K on Accuracy

The hyperparameter K denotes the number of features randomly selected at each node during the tree induction process. It must be set with an integer in the interval $[1..M]$, where M stands for the dimensionality of the feature space. This number thus controls the strength of the randomization in the feature selection process, in such a way that the smaller the value of K, the stronger the randomization. In the case where $K = 1$ for example, each split (*i.e.* the feature used as splitting criterion) of the tree structure is randomly selected among all the available features. On the contrary, where $K = M$, no randomization is introduced in the split selection and each tree is thus grown following a traditional tree induction process. In this particular case, randomization is thus introduced only through the bagging principle. The main idea of our experiments is to study RF accuracy according to hyperparameter K, on several kinds of machine learning problems. We first describe in the following subsection the datasets used. We then detail our experimental protocol and results in the next two subsections.

3.1 Datasets

The description of the 12 datasets that have been used for these experiments is summurized in Table 1. 9 of these datasets have been selected from the UCI repository [11], because they concern different machine learning issues in terms of number of classes, number of features and number of samples. Three additionnal datasets on different handwritten digit recognition problems have been used: (i) the well-known MNIST database [10] with a 85 multiresolution density feature set $(1+2\times2+4\times4+8\times8)$ built from greyscale mean values as explained in [3]; (ii) Digits and DigReject both described in [12], on which a 330-feature set has been extracted, made from three state-of-the-art kinds of descriptors, *i.e.* a 117-statistical/structural feature set [13], a 128-feature set extracted from the chaincode (contour-based) [14], and the same 85-feature set as for MNIST.

Table 1. Dataset description

Dataset	# Samples	# Features	# Classes	Dataset	# Samples	# Features	# Classes
Digits	38142	330	10	Mfeat-karhunen	2000	64	10
DigReject	14733	330	2	Mfeat-zernike	2000	47	10
Letter	20000	16	26	MNIST	60000	84	10
Madelon	2600	500	2	Musk	6597	166	2
Mfeat-factors	2000	216	10	Pendigits	10992	16	10
Mfeat-fourier	2000	76	10	Segment	2310	19	7

3.2 Experimental Protocol

Our experiments aim at studying the evolution of RF accuracy according to different values of K. For all the RF induced in our experiments, the number of trees has been set to 100. This choice is based on an experimental work presented in [3], in which it has been shown that it is a reasonable value to grow an accurate RF. Moreover our goal is not here to reach intrinsic optimal performance. First, each dataset has been randomly split into training and testing subsets, as explained in the previous section. This splitting procedure has been repeated 50 times, so that 50 different training sets and testing sets are thus available, each respectively containing two thirds and one third of the original dataset. We denote by $T_i = (Tr_i, Ts_i)$ such a split, with $i \in [1..50]$ and where Tr_i and Ts_i stand respectively for the training set and the testing set. Then, for each T_i, the Forest-RI algorithm has been run for each value of K in $[1..M]$, where M stands for the total number of features. However, mainly for computational reasons, the definition domain of K has been sampled for some of the datasets for which the size of the feature space is too large, so that values of K have been picked at regular intervals between 1 and M. In the rest of this paper we denote by M' the number of values of K that have been tested for each dataset. Table 2 summarizes the number of runs in these experiments, according to the values of K and the number of splits. Algorithm 1 summarizes the whole experimental protocol applied to each dataset. This procedure outputs a table of $50 \times M'$ error rates according to different values of K, and for each dataset. Those results are presented and discussed in the next subsection.

Table 2. Numbers of runs for evaluating RF performance according to K

Dataset	M	M' = # values of K tested for each T_i	total # of runs
Digits	330	$\frac{M}{10} + 1 = 34$	$34 \times 50 = 1700$
DigReject	330	$\frac{M}{10} + 1 = 34$	$34 \times 50 = 1700$
Letter	16	16	$16 \times 50 = 800$
Madelon	500	$\frac{M}{10} + 1 = 51$	$51 \times 50 = 2550$
Mfeat-factors	216	$\frac{M}{3} + 1 = 73$	$73 \times 50 = 3650$
Mfeat-fourier	76	76	$76 \times 50 = 3800$
Mfeat-karhunen	64	64	$64 \times 50 = 3200$
Mfeat-zernike	47	47	$47 \times 50 = 2350$
Mnist	84	$\frac{M}{2} + 1 = 43$	$43 \times 50 = 2150$
Musk	166	$\frac{M}{3} + 1 = 56$	$56 \times 50 = 2800$
Pendigits	16	16	$16 \times 50 = 800$
Segment	19	19	$19 \times 50 = 950$
Total		586	29100

3.3 Results

Table 3 presents the results obtained with the experimental protocol detailed in Algorithm 1. With this protocol, a table of $50 \times M'$ error rates with respect to K is first obtained for each dataset. These series of error rates have been averaged so that to each dataset correspond M' accuracies, *i.e.* one mean value for every K, and the standard

Algorithm 1. Experimental Protocol 1

```
INPUT: $N$: number of samples in the original dataset.
INPUT: $M$: number of features in the original dataset.
OUTPUT: $\epsilon[50][M']$: 2D table used for storing each error rate obtained with Forest-RI.
  for $i \in [1..50]$ do
    Randomly draw without replacement $\frac{2}{3} \times N$ samples from the original dataset to form
    a training subset $Tr_i$. The remaining samples form the testing subset $Ts_i$, the couple
    $(Tr_i, Ts_i)$ is denoted $T_i$.
    for each $k$ in $[1..M]$ do
      $H(k) \leftarrow$ Grow a Random Forest with the Forest-RI algorithm, with $L = 100$ and $K =$
      $k$, on the training set $Tr_i$.
      $\epsilon(i, k) \leftarrow$ Test the resulting RF on the testing set $Ts_i$.
    end for
  end for
```

deviation values have also been computed. Table 3 details some of those results according to five particular values of K. The first of those values, in the second column, corresponds to the "optimal" value of K, noted K^*, that is to say the value of K, among all the tested values, for which the maximum average accuracy has been reached. The second and third values correspond to $K = \sqrt{M}$ and $K = log_2 M + 1$ which are often used as default settings in the literature (see section 2). The two last columns present error rates for $K = 1$ and $K = M$. The number in brackets for columns 2, 3 and 4, represents the value of K that is either obtained from the experiments (K^* in column 2) or fixed for the experiments ($\sqrt{M}$ in column 3 or $log_2(M) + 1$ in column 4).

A first observation made from Table 3 is that the four particular values $K = \sqrt{M}$, $K = log_2(M) + 1$, $K = 1$ and $K = M$ rarely exactly correspond to the best parametrization of K. K^* is $\sqrt{M}$ for only 2 of the 12 datasets (Mfeat-zernike and

Table 3. Mean error rates for the different algorithms

Dataset	K^*	$K_{\sqrt{M}}$	$K_{log_2(M)+1}$	K_1	K_M
Digits	2.18 ± 0.12 (11)	2.20 ± 0.13 (18)	2.19 ± 0.12 (9)	2.61 ± 0.13	3.25 ± 0.19
DigReject	7.15 ± 0.34 (181)	7.70 ± 0.34 (18)	7.80 ± 0.35 (9)	9.02 ± 0.32	7.27 ± 0.30
Letter	4.16 ± 0.28 (3)	4.23 ± 0.23 (4)	4.30 ± 0.26 (5)	5.21 ± 0.27	7.33 ± 0.47
Madelon	17.60 ± 1.60 (261)	30.48 ± 1.94 (22)	34.54 ± 1.26 (10)	45.94 ± 1.57	18.60 ± 1.91
Mfeat-fac	3.56 ± 0.71 (10)	3.57 ± 0.58 (15)	3.58 ± 0.73 (9)	4.27 ± 0.72	4.61 ± 0.76
Mfeat-fou	16.81 ± 1 (19)	17.11 ± 1.05 (9)	17.25 ± 1.02 (7)	22.18 ± 1.19	18.66 ± 1.33
Mfeat-kar	4.30 ± 0.68 (6)	4.33 ± 0.69 (7)	4.30 ± 0.68 (6)	7.14 ± 0.83	8.38 ± 1.11
Mfeat-zer	22.26 ± 1.06 (7)	22.26 ± 1.06 (7)	22.56 ± 1.02 (5)	23.86 ± 0.90	24.87 ± 1.33
MNIST	5.06 ± 0.14 (24)	5.17 ± 0.14 (10)	5.32 ± 0.17 (8)	6.54 ± 0.17	6.54 ± 0.28
Musk	2.34 ± 0.34 (88)	2.40 ± 0.29 (13)	2.50 ± 0.26 (8)	4.03 ± 0.32	2.48 ± 0.34
Pendigits	0.97 ± 0.17 (4)	0.97 ± 0.17 (4)	1.01 ± 0.17 (5)	1.15 ± 0.18	1.50 ± 0.23
Segment	2.36 ± 0.53 (5)	2.44 ± 0.44 (4)	2.36 ± 0.53 (5)	3.23 ± 0.50	2.71 ± 0.56

Pendigits), $log_2(M)+1$ for only 2 of them (Mfeat-karhunen and Segment), and is never equal to 1 nor to M. Those settings are even sometimes quite far from the K^* value, as for the Madelon, DigRejects, MNIST and Musk datasets. One could conclude from this that it is not advised to systematically use one of those default settings of K and that it is necessary to further investigate a way to supply a better rule of parametrization. However, by examining standard deviation values and differences between results obtained for two settings of K closed from each other, we were wondering if this conclusion was correct. We can see from table 3 that they are most of the time of the same order. If we thus consider that values of K closed from each other produce classifiers statiscally equivalent in terms of accuracy, $K = \sqrt{M}$ can be considered as a good setting for K — at least close to the optimal setting — for 8 of the 12 datasets. Of course this statement needs to be experimentally investigated and proved, by performing a statistical test of significance such as the McNemar test for example. Nevertheless we think that it is reasonnable to deduce from our results that $K = \sqrt{M}$ is a good compromise for the parametrization of K.

Figure 1 presents our results as curves of averaged error rates with respect to values of K. On this figure one can first observe that all the curves exhibit the same global variation, that is to say a decreasing followed by an increasing when K increases. This confirms the primary conclusions drawn in [3], in which it has been found that the extrema values for K, *i.e.* $K = 1$ and $K = M$, are not advised to be used for building an accurate RF with Forest-RI. However, this common behavior strongly differs from a dataset to another in a more detailed analysis of the curves. We can distinguish three trends of variation in these 12 diagrams: for 5 of them (Mfeat-factors, Mfeat-fourier, Mfeat-karhunen, Mfeat-zernike and Digits) the minimum error rate is reached for a small value of K (marked with circles in the figure) and the increase of the curves from this point till $K = M$ is monotonical and almost linear; for 4 of them (Letter, Mnist, Pendigits and Segment) the trend is almost the same but with a more parabolical shape; for 3 of them (DigReject, Madelon and Musk) this increase of the curves is quasi null, and the minimum error rate is reached for a larger value of K (*i.e.* greater than $\frac{M}{2}$). These different behaviours are quite difficult to explain only with some characteristics of the datasets such as the number of classes or the number of features. Geurts et al. in [6] suggest that the nature of the features could explain some particularity of their Extra-Trees behavior for some particular cases. They conjecture that the more features that are irrelevant, the larger the value of K^*, since *a higher value of K would lead to a better chance of filtering out the irrelevant variables*. This statement has led us to focus on the relevancy of features for explaining accuracy variation according to values of K. For that purpose we have decided to evaluate the relevancy of each feature by measuring the information gain with respect to the class. In general terms, the expected information gain is the change in information entropy from a prior state to a state that takes some information [15]. It is often used in decision tree induction as criterion for node split selection. For our experiments, information gain has been measured for all the features on each Tr_i of each dataset. In that way, $50 \times M$ values of information gain have been computed for each dataset. Figure 2 synthesizes those results as curves of cumulative number of features with respect to information gain values, so that each dot of the curves indicates the number of features for which the information gain is smaller

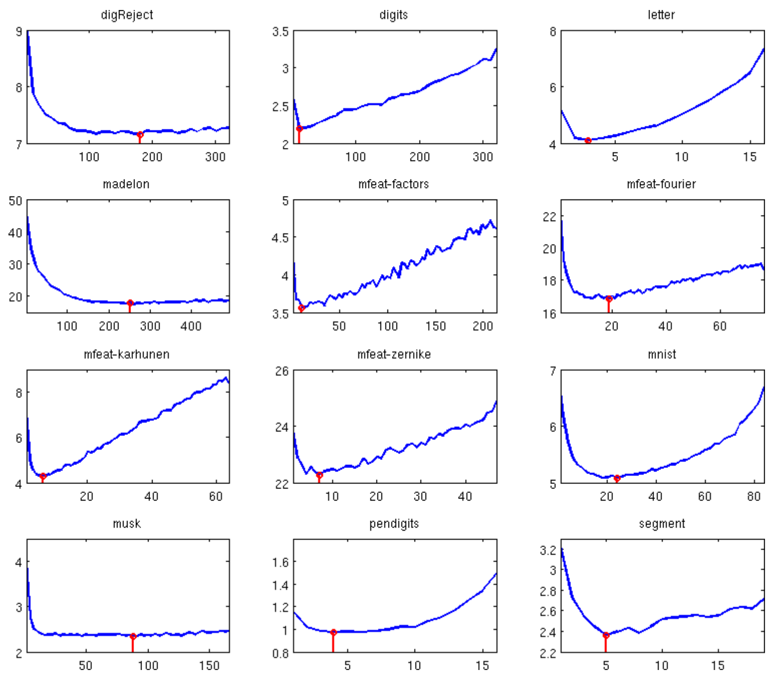


Fig. 1. Mean error rates with respect to values of K. The minimum error rates are marked on each diagram.

than or equal to the corresponding value on the x-axis. This representation allows to simultaneously observe the relevancy of all the features and gives an idea of how many of them are irrelevant. One can observe on this figure that values of information gain are globally greater (typically higher than 0.1) for datasets for which values of K^* are small regarding to M, as this is the case for exemple for Digits, Mfeat-Factors and Mfeat-Karhunen for which K^* is lower than $\frac{M}{10}$. On the contrary, for the three datasets for which K^* is higher than $\frac{M}{2}$ (DigReject, Madelon and Musk), the information gain values are always lower than about 0.1. This seems to prove that relevancy of features strongly explains the accuracy variation according to values of K.

The choice of K actually leans on a compromise between two needs : (i) to force, via randomness, the tree induction process to diversify choices of splitting criteria in order to induct trees different from each other (ii) to choose relevant features for splitting criteria in order to induct trees performant enough. A too strong randomization of the split selection produces trees that globally do not suit the problem enough, while not randomizing ”enough” creates trees that tend to overfit the training data, and thus make them be similar from each other in terms of predictions. K acts thus as a trade-off for balancing performance and diversity of trees in the ensemble. However we believe that the impact on performance of the ”amount” of randomization used in the split selection process, strongly depends on the global relevancy of features. If there are too few relevant features, the randomization will rapidly make the tree accuracy decrease, and

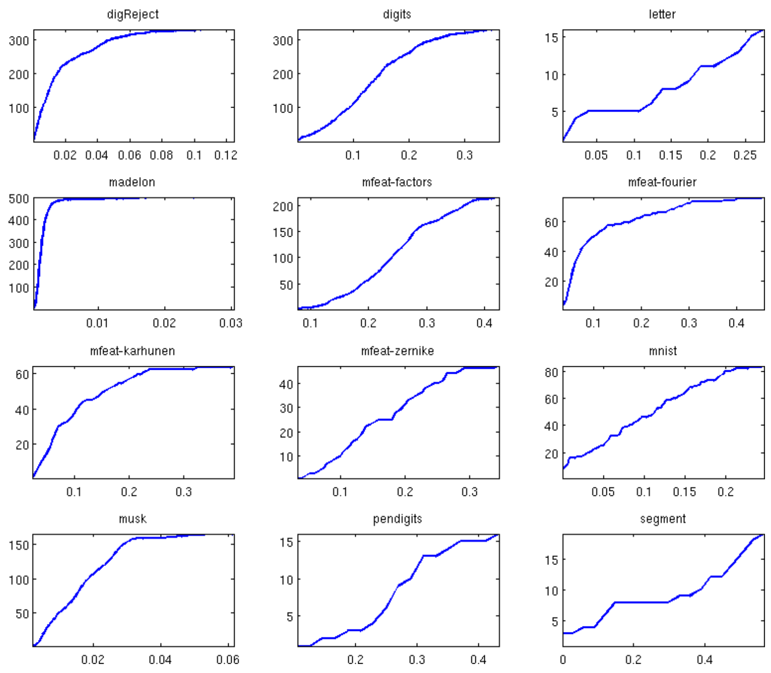


Fig. 2. Cumulative number of features with respect to information gain values

thus rapidly deteriorate the "individual performance of the trees versus the ensemble diversity" trade-off. On the contrary a large amount of strongly relevant features facilitates the overfitting of trees and weakens the randomization effects in split selection. We believe that this is the reason why the three datasets for which values of information gain are small (DigRejects, Madelon and Musk), present error rate curves that do not significantly raise for increasing values of K. Consequently we think that the relevancy of features is an important property that should be taken into account for determining a parametrization rule for the Forest-RI algorithm.

4 Conclusions

Investigations on RF parametrization have been presented in this paper, that have focused on the number K of features randomly selected at each node during the tree induction. This hyperparameter allows to control the strengh of the randomization in the split selection, in such a way that the smaller the value of K, the stronger the randomization. In this work several experiments have been led with the Forest-RI algorithm on different machine learning datasets and with different settings of K, in order to study its influence on RF performance. We have firstly shown that default settings traditionally used in the literature do not allow to produce the best possible RF in terms of accuracy, in a majority of cases. However our results illustrate that one of them, *i.e.* $K = \sqrt{M}$

with M being the dimensionality of the feature space, is a reasonnable setting to induct near-optimal RF. This statement should obviously be experimentally confirmed through the use of statistical test of significance such as the McNemar test for determining whether or not significant improvement can be made with an optimal setting of K in comparison with $K = \sqrt{M}$. In a second part of this experimental work we have focused on the relevancy of features to determine whether or not it could explain the accuracy variation according to K. We have shown that this property is crucial for finding the best setting of K since it can strongly modify the randomization effect of the random split selection procedure. As a consequence we think that the relevancy of features is an important property that should be taken into account for determining a parametrization rule for the Forest-RI algorithm. Our future works will focus on this open issue.

References

1. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
2. Ho, T.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
3. Bernard, S., Heutte, L., Adam, S.: Using random forests for handwritten digit recognition. In: International Conference on Document Analysis and Recognition, pp. 1043–1047 (2007)
4. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
5. Breiman, L.: Consistency of random forests and other averaging classifiers. Technical Report (2004)
6. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning 36(1), 3–42 (2006)
7. Latinne, P., Debeir, O., Decaestecker, C.: Limiting the number of trees in random forests. In: Kittler, J., Roli, F. (eds.) MCS 2001. LNCS, vol. 2096, pp. 178–187. Springer, Heidelberg (2001)
8. Rodriguez, J., Kuncheva, L., Alonso, C.: Rotation forest: A new classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(10), 1619–1630 (2006)
9. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Chapman and Hall/Wadsworth, Inc., New York (1984)
10. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
11. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
12. Chatelain, C., Heutte, L., Paquet, T.: A two-stage outlier rejection strategy for numerical field extraction in handwritten documents. In: International Conference on Pattern Recognition, Honk Kong, China, vol. 3, pp. 224–227 (2006)
13. Heutte, L., Paquet, T., Moreau, J., Lecourtier, Y., Olivier, C.: A structural/statistical feature based vector for handwritten character recognition. Pattern Recognition Letters 19(7), 629–641 (1998)
14. Kimura, F., Tsuruoka, S., Miyake, Y., Shridhar, M.: A lexicon directed algorithm for recognition of unconstrained handwritten words. IEICE Transaction on Information and System E77-D(7), 785–793 (1994)
15. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182 (2003)

Ensembles of One Class Support Vector Machines

Albert D. Shieh[1] and David F. Kamm[2]

[1] Department of Statistics, Harvard University,
Cambridge, MA 02138, USA
shieh@fas.harvard.edu

[2] Department of Computer Science, Stanford University,
Stanford, CA 94305, USA
dkamm@cs.stanford.edu

Abstract. The one class support vector machine (OCSVM) is a widely used approach to one class classification, the problem of distinguising one class of data from the rest of the feature space. However, even with optimal parameter selection, the OCSVM can be sensitive to overfitting in the presence of noise. Bagging is an ensemble method that can reduce the influence of noise and prevent overfitting. In this paper, we propose a bagging OCSVM using kernel density estimation to decrease the weight given to noise. We demonstrate the improved performance of the bagging OCSVM on both simulated and real world data sets.

1 Introduction

In binary classification, it is typically assumed that training data is available for both classes. However, in some real world applications, there is little or no data available for one of the classes. For example, in the diagnosis of rare diseases, there are many healthy patients but few sick patients to collect data from. Therefore, one class is represented well (positive class), but the other class is not represented well or at all (negative class). One class classification is the partially unsupervised learning problem of training on positive data only and distinguishing the positive class from the rest of the feature space, which comprises all possible negative classes. One class classifiers decide to either accept or reject a given point into the positive class [13].

There are two main types of one class classifiers. Density methods estimate the probability distribution of the positive class and accept points that have a high probability of belonging to the positive class. Boundary methods estimate a boundary that encloses the positive class and accept points inside the boundary. Although density methods provide information about the entire structure of the positive class, boundary methods often perform better since they solve a fundamentally easier problem [13]. In particular, the one class support vector machine (OCSVM), a boundary method adapting the support vector machine (SVM) to one class classification [12,11], has become one of the most widely used one class classifiers.

Although the OCSVM has been applied widely, the estimated boundary can be sensitive in practice [10]. When the training data is noisy and contains many outliers near or in the negative class, the OCSVM will estimate a large boundary that encloses areas of the feature space where the positive class has low density, resulting in many false positives [6]. This can be highly problematic for many applications where the number of false positives must be kept to a minimum. For example, the accidental diagnosis of a sick patient as healthy may result in death. Optimal parameter selection can tighten the estimated boundary to a certain extent [15], but ideally the OCSVM should be modified in order to decrease the influence of outliers on the estimated boundary.

One way of viewing the sensitivity of the OCSVM to outliers is that the OCSVM is unstable. Although the estimated boundary robustly encloses the positive class, the introduction of outliers can arbitrarily expand the estimated boundary. Bagging is an ensemble method that combines multiple unstable classifiers trained on resampled data to produce an improved classifier [3]. Bagging has been applied to other unstable one class classifiers such as one class decision trees [9] with success. Other types of ensemble methods, such as combining different one class classifiers [14] and training on different sets of features [8], have been applied to the OCSVM with success. However, since the OCSVM has been traditionally viewed as stable, bagging OCSVM has not been explored to our knowledge.

In this paper, we propose a bagging OCSVM using weights determined by kernel density estimation. Our bagging OCSVM combines the advantages of density methods, boundary methods, and bagging. The OCSVM is still used to estimate a boundary, but kernel density estimation is used to find outliers and bagging is used to decrease the influence of outliers on the estimated boundary. Our bagging OCSVM is inspired by the recent success of combining density and boundary methods [5]. We demonstrate that the bagging OCSVM tightens the estimated boundary and reduces false positives on both simulated and real world data sets. The rest of the paper is organized as follows. In Section 2, we describe the OCSVM. In Section 3, we describe the bagging OCSVM. In Section 4, we experimentally compare the normal and bagging OCSVM. In Section 5, we give our final remarks.

2 One Class Support Vector Machines

The OCSVM, as formulated in [11], estimates a set that encloses most of a given sample of m points $\{\mathbf{x}_i\}_{i=1}^m, \mathbf{x}_i \in \mathbb{R}^d$. Each point $\mathbf{x}_i$ is transformed by a map $\phi : \mathbb{R}^d \to \mathcal{K}$ from the feature space $\mathbb{R}^d$ into a high dimensional kernel space $\mathcal{K}$ generated by the kernel $k(\mathbf{x}, \mathbf{y})$. The kernel corresponds to an inner product in the kernel space through $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$. The OCSVM finds a hyperplane in the kernel space that separates the data from the origin with maximum margin. If no such hyperplane exists, slack variables ξ_i allow for some points to be within the margin (outliers) and a free parameter $\nu \in [0, 1]$ controls the cost of such violations. In general, ν provides an upper bound on the fraction of outliers [11]. The hyperplane in the kernel space induces a nonlinear surface in the feature space.

More precisely, the OCSVM solves the quadratic program

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi},\rho} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i - \rho \\ \text{s.t.} \quad & \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle \geq \rho - \xi_i,\ i = 1,\ldots,m \\ & \xi_i \geq 0,\ i = 1,\ldots,m \end{aligned} \tag{1}$$

where $\mathbf{w}$ is the normal vector to the hyperplane and ρ is the margin. The quadratic program can be solved efficiently by sequential minimal optimization (SMO) of its dual form [11]. The decision function

$$g(\mathbf{x}) = \operatorname{sign}(\langle \mathbf{w}, \phi(\mathbf{x})\rangle - \rho) \tag{2}$$

determines whether a given point $\mathbf{x}$ is in (positive) or out (negative) of the estimated set.

In practice, a Gaussian kernel

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right) \tag{3}$$

with a width parameter σ is used and is the only kernel that has been successfully applied to the OCSVM [11]. The Gaussian kernel maps the data into the same orthant in the kernel space, justifying the principle of separating the data from the origin. The OCSVM with a Gaussian kernel is equivalent to the support vector domain description (SVDD) [12], which finds a hypersphere in the kernel space that encloses the data with minimum volume.

Optimal selection of σ is critical to the performance of the OCSVM, since it controls the shape of the estimated boundary. When σ is not selected properly and distances are inhomogeneous, the estimated boundary is often elongated and encloses large, empty areas of the feature space. Selecting σ using methods such as kernel whitening [15] can tighten the estimated boundary to a certain extent. However, when there are many outliers in the data, tuning σ alone is not sufficient to create a compact estimated boundary [6]. Therefore, the OCSVM should be modified in order to tighten the estimated boundary and exclude outliers.

3 Weighted Bagging

Bagging is a popular ensemble method that trains each classifier in the ensemble on a resampled version of the training data [3]. Given training data of m points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{m}$, a bootstrap sample $\mathcal{Y}$ is generated by drawing m points randomly with replacement from $\mathcal{X}$ with probability weight $w(i)$ for point $\mathbf{x}_i$. In classical bagging, all points are given the same probability weight $w(i) = 1/m$. Each bootstrap sample is the same size as the training data, but some points from the training data can either be left out or repeated multiple times. An ensemble of n

classifiers is formed by training a classifier on n bootstrap samples. New points are typically classified by a simple majority vote of the ensemble.

More precisely, the bagging algorithm is:

1. Generate n bootstrap samples of m points $\{\mathcal{Y}_j\}_{j=1}^n$ from $\mathcal{X}$ with probability weights $w(i)$.
2. For $j = 1, \ldots, n$, train a classifier g_j on the bootstrap sample $\mathcal{Y}_j$.
3. Classify new points using the majority vote of the ensemble $\{g_j\}_{j=1}^n$.

Bagging is only useful when the classifier is unstable and small changes to the training data in the bootstrap samples can create large changes in the classifier [1]. The OCSVM is unstable in the sense that the estimated boundary expands greatly in the presence of outliers. However, the OCSVM is also stable in the sense that the estimated boundary always encloses the positive class.

Directly applying bagging to the OCSVM is not useful since the OCSVM is stable for the majority of the training data. Bagging only tightens the estimated boundary when outliers are excluded from the bootstrap samples. However, since all points are given the same probability weight, outliers are likely to be included in many of the bootstrap samples. Bagging will still exclude some outliers from the bootstrap samples by chance, but the estimated boundary will not be very robust. Since we want to exclude outliers from the bootstrap samples, we should give outliers lower probability weight. We propose to give probability weights to points based on how close they are to the positive class.

Kernel density estimation is a popular nonparametric method to estimate the probability distribution of the training data [7]. The kernel density estimator is a sum of Gaussian kernels at each point in the training data

$$f(\mathbf{x}) = \sum_{j=1}^{m} \frac{1}{(2\pi)^{d/2}\sigma^d m} k(\mathbf{x}, \mathbf{x}_j). \tag{4}$$

We could use the kernel density estimate as the probability weight $w(i) = f(\mathbf{x}_i)$ for point $\mathbf{x}_i$. However, kernel density estimation in multiple dimensions can be unreliable. Therefore, a more suitable probability weight would be a measure of how well the kernel density estimator fits the training data.

We use an iterative method from [4] based on cross validation of a weighted kernel density estimator to determine the probability weights. Instead of boosting up probability weights for outliers, we boost them down. Initially, all points are given the same probability weight. At iteration k, given the probability weights $w_k(i)$, the weighted kernel density estimator is

$$f_k(\mathbf{x}_i) = \sum_{j=1}^{m} \frac{w_k(i)}{(2\pi)^{d/2}\sigma^d} k(\mathbf{x}_i, \mathbf{x}_j) \tag{5}$$

and the leave one out weighted kernel density estimator is

$$f_k'(\mathbf{x}_i) = \sum_{j=1}^{m} \frac{w_k(i)}{(2\pi)^{d/2}\sigma^d} k(\mathbf{x}_i, \mathbf{x}_j) I(j \neq i). \tag{6}$$

The probability weights are updated by adding the log odds ratio of the weighted kernel density estimate to the leave one out weighted kernel density estimate.

More precisely, the probability weight algorithm is:

1. Initialize with uniform probability weights by $w_0(i) = 1/m$.
2. For $k = 1, \ldots, n$, update the probability weights by

$$w_k(i) = w_{k-1}(i) + \log \left(\frac{f_{k-1}(\mathbf{x}_i)}{f'_{k-1}(\mathbf{x}_i)} \right).$$

3. Invert and normalize the final probability weights by

$$w(i) = \left(\frac{1}{w_n(i)} \right) / \left(\sum_{j=1}^{m} \frac{1}{w_n(j)} \right).$$

The final probability weights will be low for points in the negative class and high for points in the positive class.

4 Experiments

We compared the normal and bagging OCSVM on both simulated and real world data sets. All experiments were performed in MATLAB on a standard personal computer. We used the SMO algorithm from [11] as implemented in the LIBSVM library to solve the OCSVM. We selected σ for each data set using a simple grid search as in [6] to maximize the number of negative points outside the estimated boundary plus the number of positive points inside the estimated boundary. We used $n = 10$ samples in bagging and $k = 5$ iterations in determining the probability weights for all data sets. We used the same ν and σ values for each individual OCSVM in the ensemble as the normal OCSVM. The bagging OCSVM was significantly more computationally intensive than the normal OCSVM due to the kernel density estimation step, but was still fast enough to be used practically. Typical runtimes for the bagging OCSVM were less than a minute.

First, we evaluated the normal and bagging OCSVM on three simulated data sets similar to those used in [6]:

– **Square noise** contains 450 points and 2 features. First, 400 points were drawn randomly from the square $\{(x, y) : x \in [0.4, 2.6], y \in [0.4, 0.6] \cup [2.4, 2.6]\} \cup \{(x, y) : x \in [0.4, 0.6] \cup [2.4, 2.6], y \in [0.4, 2.6]\}$. Next, 50 points of noise were drawn randomly from the area $\{(x, y) : x \in [0, 3], y \in [0, 3]\}$. We set $\nu = 1/9$ and $\sigma = 0.35$.
– **Line noise** contains 450 points and 2 features. First, 400 points were drawn randomly from the line $\{(x, y) : x = y, x \in [0, 3], y \in [0, 3]\}$. Next, 50 points of noise were drawn randomly from the area $\{(x, y) : x \in [0, 3], y \in [0, 3]\}$. We set $\nu = 1/9$ and $\sigma = 0.35$.

– **Sphere** contains 450 points and 2 features. All 450 points were drawn from a bivariate Gaussian distribution with mean $(1.5, 1.5)$ and variance 0.1. We set $\nu = 1/10$ and $\sigma = 2$.

The estimated boundaries of the normal and bagging OCSVM are shown in Figure 1. On the Square noise and Line noise data sets, the bagging OCSVM

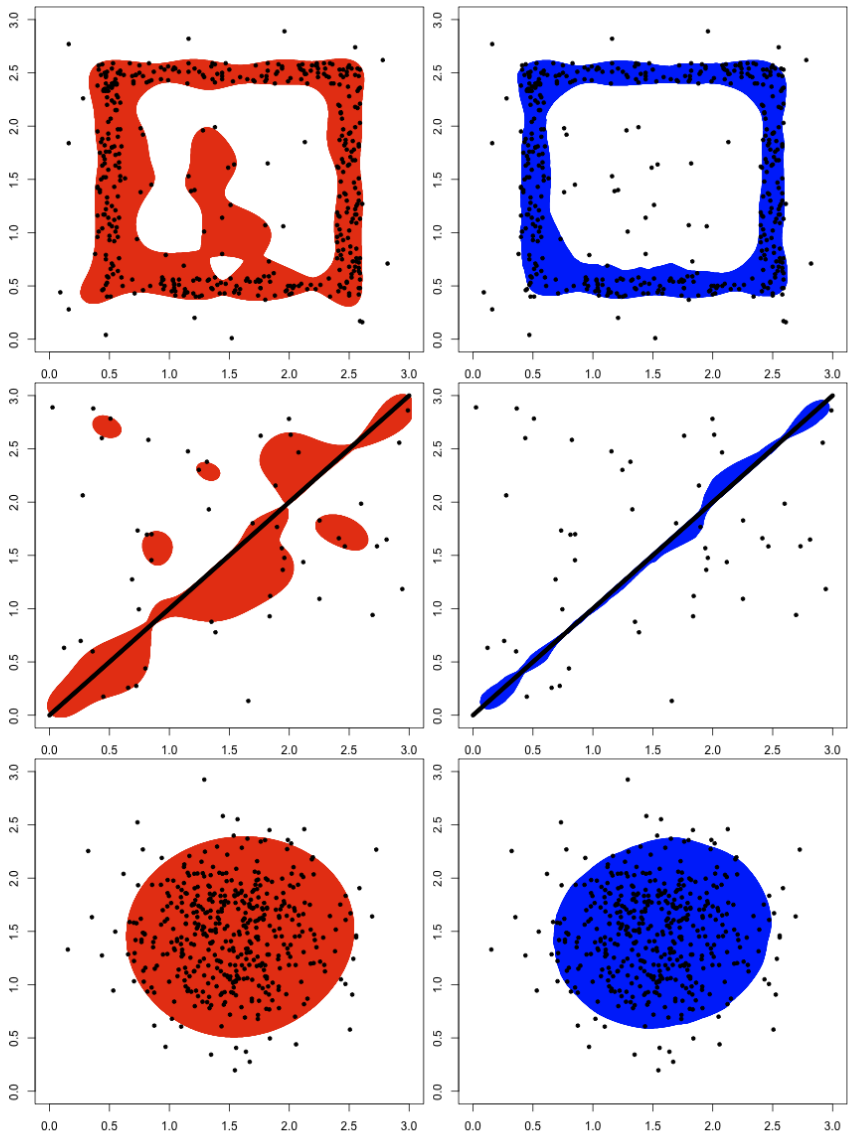


Fig. 1. Estimated boundaries of the normal (left) and bagging (right) OCSVM on the Square noise (top), Line noise (middle), and Sphere (bottom) data sets

performs much better than the normal OCSVM. The estimated boundary of the normal OCSVM is influenced by outliers and encloses large areas of the feature space with low density, including disconnected areas of the feature space. On the other hand, the estimated boundary of the bagging OCSVM tightly encloses the shapes of the positive class in the feature space. Therefore, the bagging OCSVM appears to be less sensitive than the normal OCSVM to noise. On the Sphere data set, the bagging OCSVM performs similarly to the normal OCSVM. Therefore, the bagging OCSVM does not appear to arbitrarily tighten the estimated boundary. Although we chose the ν values optimally according to the proportion of noise, we found that adjusting the ν values only shrunk or expanded the estimated boundaries uniformly for both the normal and bagging OCSVM.

In order to determine whether the probability weights alone are sufficient to eliminate outliers without bagging, we removed 50 points with the lowest probability weights from the Square noise and Line noise data sets and trained a normal OCSVM. The amount of outliers removed corresponds to the amount of added noise. The estimated boundaries of the normal OCSVM are shown in Figure 2. Although the estimated boundaries of the normal OCSVM improved significantly after outlier removal, the probability weights are not necessarily reliable for outlier detection. In regions of the feature space where the data is sparse, the probability weights are low, so thresholding can lead to discontinuities in the estimated boundary. In particular, the estimated boundary for the Square noise data set contains a gap in the top side. Therefore, sampling from a weighted distribution in bagging appears to be important for averaging out sparse regions of the data.

Next, we evaluated the performance of the normal and bagging OCSVM on three real world data sets from the UCI Machine Learning Repository:

– **USPS** contains 256 features and 2651 points (821 positive class, 1830 negative class). The data set was randomly partitioned into a training data set

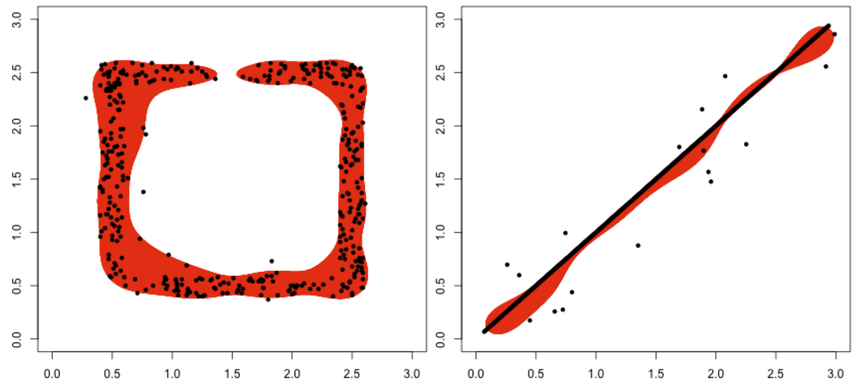


Fig. 2. Estimated boundaries of the normal OCSVM on the Square noise (left) and Line noise (right) data sets after outlier removal

Table 1. Performance of the normal and bagging OCSVM on the USPS, Breast cancer, and Ionosphere data sets

Data set	FPR	TPR		
		Normal	Bagging	Difference
USPS	0%	45.2%	49.7%	+4.5%
	1%	61.6%	65.0%	+3.4%
	5%	75.7%	78.5%	+2.8%
	10%	89.8%	92.7%	+2.9%
Breast cancer	0%	67.2%	88.9%	+21.7%
	1%	82.0%	89.8%	+7.8%
	5%	86.1%	90.2%	+4.1%
	10%	93.0%	92.2%	−0.8%
Ionosphere	0%	8.8%	18.4%	+9.6%
	1%	28.0%	37.6%	+9.6%
	5%	58.4%	60.0%	+1.6%
	10%	87.2%	90.4%	+3.2%

of 644 points and a test data set of 2007 points (177 positive class, 1830 negative class). We set $\sigma = 1$.
- **Breast cancer** contains 10 features and 683 points (444 positive class, 239 negative class). The data set was randomly partitioned into a training data set of 200 points and a test data set of 483 points (244 positive class, 239 negative class). We set $\sigma = 1$.
- **Ionosphere** contains 34 features and 351 points (225 positive class, 126 negative class). The data set was randomly partitioned into a training data set of 100 points and a test data set of 251 points (125 positive class, 126 negative class). We set $\sigma = 16$.

We used the true positive rate (TPR) and false positive rate (FPR) as our performance metrics. Since ν is an upper bound on the FPR, we varied ν in order to control the FPR. However, varying ν was not sufficient to compute a full receiver operating characteristic (ROC) curve since the FPR was always below 25%. In real world applications of one class classification, the target FPR is typically very low since there are large consequences for false positives. Therefore, we compared the TPR of the normal and bagging OCSVM at four typical target FPRs of 0%, 1%, 5%, and 10% as in [8]. The TPRs of the normal and bagging OCSVM are shown in Tables 1 and 2.

The bagging OCSVM achieves higher TPRs than the normal OCSVM for almost all target FPRs on all data sets, suggesting that real world data sets are noisy enough that tightening the estimated boundary is important. The performance improvement of the bagging OCSVM is highest for low target FPRs. For a target FPR of 0%, the difference in TPR for the bagging OCSVM ranges from +4.5% to +21.7%. As the target FPR increases, the performance of the bagging OCSVM approaches that of the normal OCSVM, probably because tightening the estimated boundary becomes less important than enclosing all of the positive class. For a target FPR of 10%, the difference in TPR for the

Table 2. Performance of the normal and bagging OCSVM on the noisy versions of the USPS, Breast cancer, and Ionosphere data sets

Data set	FPR	TPR		
		Normal	Bagging	Difference
USPS	0%	32.2%	38.4%	+6.2%
	1%	48.6%	53.1%	+4.5%
	5%	68.4%	73.4%	+5.0%
	10%	81.9%	84.7%	+2.8%
Breast cancer	0%	56.1%	83.6%	+27.5%
	1%	59.8%	86.9%	+27.1%
	5%	66.8%	87.3%	+20.5%
	10%	78.3%	87.8%	+9.5%
Ionosphere	0%	0.0%	11.2%	+11.2%
	1%	25.6%	36.0%	+10.4%
	5%	56.8%	58.4%	+1.6%
	10%	59.2%	60.0%	+0.8%

bagging OCSVM ranges from −0.8% to +3.2%. Therefore, the bagging OCSVM appears to be well suited for applications of one class classification that require a low target FPR.

In order to further evaluate the robustness, noisy versions of the data sets were generated by randomly swapping 25% of the points in the training data set with points in the test data set. The performance improvement of the bagging OCSVM was even larger on the noisy versions of the data sets, suggesting that tightening the estimated boundary is especially important in the presence of noise. The noisy versions of the training data sets contained points from both the positive class and the negative class, which probably expanded the estimated boundaries of the normal OCSVM. The performance of the bagging OCSVM is not as sensitive to noise since the bagging OCSVM decreases the weight given to points far from the positive class. Therefore, the bagging OCSVM appears to be well suited for unlabeled training data.

5 Conclusion

In this paper, we proposed a bagging OCSVM using weights determined by kernel density estimation to tighten the estimated boundary of the normal OCSVM, which can be sensitive to noise. We demonstrated that the estimated boundary of the bagging OCSVM fits the shape of the positive class well on three simulated data sets and that the bagging OCSVM achieves significantly higher TPRs than the normal OCSVM on three real data sets at common target FPRs. The bagging OCSVM is especially useful for applications of one class classification that require low target FPRs, such as the diagnosis of rare diseases.

Acknowledgments. The authors would like to thank the anonymous reviewers, whose comments helped improve the manuscript.

References

1. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36, 105–139 (1999)
2. Bicego, M., Figueiredo, M.A.T.: Soft clustering using weighted one-class support vector machines. Pattern Recognition 42, 27–32 (2009)
3. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
4. Di Marzio, M., Taylor, C.C.: Boosting kernel density estimates: a bias reduction technique? Biometrika 91, 226–233 (2004)
5. Hempstalk, K., Frank, E., Witten, I.H.: One-class classification by combining density and class probability estimation. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 505–519. Springer, Heidelberg (2008)
6. Hoffmann, H.: Kernel PCA for novelty detection. Pattern Recognition 40, 863–874 (2007)
7. Parzen, E.: On estimation of a probability density function and mode. Annals of Mathematical Statistics 33, 1065–1076 (1962)
8. Perdisci, R., Gu, G., Lee, W.: Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems. In: 6th IEEE International Conference on Data Mining, pp. 488–498. IEEE Press, New York (2006)
9. Li, C., Zhang, Y.: Bagging one-class decision trees. In: 5th International Conference on Fuzzy Systems and Knowledge Discovery, pp. 420–423. IEEE Press, New York (2008)
10. Roth, V.: Kernel fisher discriminants for outlier detection. Neural Computation 18, 942–960 (2006)
11. Scholköpf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation 13, 1443–1471 (2001)
12. Tax, D.M.J., Duin, R.P.W.: Support vector domain description. Pattern Recognition Letters 20, 1191–1999 (1999)
13. Tax, D.M.J.: One-class classification. Ph.D thesis, Delft University of Technology (2001)
14. Tax, D.M.J., Duin, R.P.W.: Combining one-class classifiers. In: Kittler, J., Roli, F. (eds.) MCS 2001. LNCS, vol. 2096, pp. 299–308. Springer, Heidelberg (2001)
15. Tax, D.M.J., Juszczak, P.: Kernel whitening for one-class classification. In: Lee, S.-W., Verri, A. (eds.) SVM 2002. LNCS, vol. 2388, pp. 40–52. Springer, Heidelberg (2002)

Disturbing Neighbors Ensembles for Linear SVM

Jesús Maudes, Juan J. Rodríguez, and César García-Osorio

University of Burgos, Spain
{jmaudes,jjrodriguez,cgosorio}@ubu.es

Abstract. Ensembles need their base classifiers do not always agree for any prediction (*diverse* base classifiers). Disturbing Neighbors ($\mathcal{DN}$) is a method for improving the diversity of the base classifiers of any ensemble algorithm. $\mathcal{DN}$ builds for each base classifier a set of extra features based on a 1-Nearest Neighbors (1-NN) output. These 1-NN are built using a small subset of randomly selected instances from the training dataset. $\mathcal{DN}$ has already been proved successfully on unstable base classifiers (i.e. decision trees). This paper presents an experimental validation on 62 UCI datasets for standard ensemble methods using Support Vector Machines (SVM) with a linear kernel as base classifiers. SVMs are very stable, so it is hard to increase their diversity when they belong to an ensemble. However, experiments will show that $\mathcal{DN}$ usually improves ensemble accuracy and base classifiers diversity.

Keywords: SVM, Ensembles, Diversity, Disturbing Neighbors, Kappa-Error Movement Diagrams.

1 Introduction

An ensemble is a combination scheme of individual predictors called base classifiers. The success of an ensemble requires both accuracy and diversity of its base classifiers. Diversity represents how different are the predictions of the base classifiers. If base classifiers always agree there would no be any difference between using only one base classifier or several combined by an ensemble method. So the power of using a set of base classifiers relies on the possibility that some of them can correct a wrong prediction of others.

It is very usual to obtain these base classifiers in an ensemble using the same algorithm, so in this situation the training process performed by the ensemble is the main source of diversity. Bagging [1] diversity comes from randomly picking different instances for training each base classifier. The Random Subspaces method [2] chooses different subsets of attributes for training each base classifier. Boosting [3] trains iteratively the set of base classifiers, modifying the weights of instances to train the current classifier. These new weights are computed from the training error on the previous base classifier, so each new base classifier becomes more specialized in instances that have been misclassified before. Sometimes base classifiers methods are very stable and the ensemble training algorithm is not enough to provide the desired level of diversity.

SVM (Support Vector Machine) [4] computes an optimal hyperplane that separates the input space in two regions corresponding to classes of a two-class dataset (i.e., it maximizes the margin). If the dataset is not linearly separable the hyperplane can be constructed without such problem in the feature space given by a *kernel* function. When no kernel is used (i.e., SVM is an hyperplane in the input space) it is said that the kernel is linear. Linear kernel is more appropriate for linearly separable datasets. However, it is also an interesting option for other datasets since it is the fastest and there exists optimized implementations (e.g., [5]). Besides, if complete separation is not possible, slack variables can be introduced to allow training errors. Thus linear kernel is a very competitive choice when numerous SVMs have to be constructed, as it happens with ensembles. For that reason this paper is focused only on linear SVM. Nevertheless, we think that some of its conclusions could be valid for other kernels.

It is known that SVM is a very stable classifier, thus it is hoped that ensembles of SVM will benefit from strategies contributing to increase their diversity.

Disturbing Neighbors ($\mathcal{DN}$) have been used successfully to improve diversity on forests [6]. $\mathcal{DN}$ uses a 1-Nearest Neighbor (1-NN) classifier to build a set of extra features that are added to the training dataset of each base classifier. This 1-NN classifier is different for each base classifier. The built features are the 1-NN prediction plus a boolean set of features indicating which is the nearest neighbor. The original training dataset is transformed into an augmented dataset, which is different for each base classifier, independently of the ensemble schema in which it is going to be used.

Unlike SVM, decision trees are very sensitive to small changes in the training dataset. The motivation of this paper is to test $\mathcal{DN}$ within SVM ensembles to see if accuracy and diversity increase, just like in forests, despite SVM stability. Although $\mathcal{DN}$ are introduced in [6], the method is described here for the sake of self-containment.

The paper is organized as follows. Section 2 describes the Disturbing Neighbor method. Section 3 analyses experimentally our method applied to state of art representative ensembles of SVMs. Section 4 concludes.

2 Method

The $\mathcal{DN}$ method (see Fig. 1) works on each base classifier as follows:

1. m instances are randomly selected from the training dataset to build a 1-NN classifier. The value m uses to be very small ($m = 10$ in our experiments).
2. Dimensions used to compute Euclidean distances in the 1-NN classifier are also randomly selected. At least 50% of attributes are selected.
3. Then $m+1$ new features are appended to training dataset. One of the added features is the class predicted by the 1-NN classifier for each instance x, and the other m are boolean features, all set to false except the one corresponding to the nearest neighbor of the instance.
4. The base classifier is trained using the original features plus the new $m+1$ features.

Thus normal base classifiers training process is altered or *disturbed* by adding these new features from the 1-NN classifier. That is why the method is called *Disturbing Neighbors*. Randomness increases diversity and it is due to:

– The neighbors used in each 1-NN classifier are selected randomly. Therefore, their predictions and the boolean features are different for each base classifier.

```
**Function** $\mathcal{DN}$**-BaseClassifierTrainer**
**input**  : $D$: Training Dataset with $l$ features and $n$ instances,
           $m$: Small integer,
           $BC_T$: Training algorithm of a base classifier $BC$
**output**: A classifier trained using a $\mathcal{DN}$ variant of $BC$ base method that can be used as
           base classifier into an ensemble method
**variables:**
$RndDimensions$: Array $[1..l]$ of booleans
$RndNeighbors$: Array $[1..m]$ of instances from $D$
$D'$ : Augmented Dataset, initially empty
**begin**
    Randomly set to $True$ more than $l/2$ elements of $RndDimensions$, the rest are set
    to $False$ ;
    Randomly fill $RndNeighbors$ with $m$ instances belonging to $D$ ;
    **forall** $x \in D$ **do**
        $x' \leftarrow x$ ;
        $i \leftarrow$ NearestNeighbor( $x$, $RndNeighbors$, $RndDimensions$) ;
        Append $m$ new boolean features into $x'$, all them with $False$ value except the
        one in $i$ position that is set to $True$;
        $p \leftarrow$ class of $RndNeighbors\,[i]$ ;
        Append $p$ as a new feature of $x'$ ;
        Insert $x'$ into $D'$ dataset ;
    **end**
    Train a $BC$ classifier using $D'$ and $BC_T$ ;
    Return $BC$;
**end**

**Function NearestNeighbor**
**input**  : $x$:training dataset instance,
           $Neighbors$: Array $[1..m]$ of instances,
           $BooleanMask$: Array $[1..l]$ of Boolean
**output**: i:integer indicating the nearest neighbor
**begin**
    Get the Nearest Neighbor to $x$ in $Neighbors$ computing the Euclidean Distance
    using only dimensions set to $True$ in $BooleanMask$ ;
    Return the index in $Neighbors$ of the 1-NearestNeighbor ;
**end**
```

Fig. 1. *Disturbing Neighbor* Base Classifier Training. 1-Nearest Neighbor function has been specified separately in order to remark that (i) it only returns the nearest neighbor index (not the predicted class), and (ii) distance is computed taking into account only a random subset from the original features.

– The dimensions used for computing the Euclidean distances are picked randomly as well, so even if two base classifiers have almost the same m neighbors, the 1-NN predictions and the boolean features could be different.

The value m is very small since the aim is not to create a very accurate 1-NN classifier, but a different one each time. The m boolean features can be taken into account by the resulting SVM, so it is expected that each SVM will be different. This deserves further explanation, our intuition is that if r is a Voronoi region corresponding to one of the m neighbors, d_j is the binary attribute indicating whether an instance is located at r, and c_j is the SVM coefficient for such attribute, then the more populated is r by instances belonging to one of the classes the SVM separates, probably the greater would be the absolute value of c_j.

Because 1-NN class predictions are nominal, they have to be transformed into binary features to be computed by SVM training algorithm. These binary features also divide the input space in regions whose number is equal or less than the number of classes. Actually, each region resulting from this new division is the union of the Voronoi regions corresponding to neighbors that share the same class. So again, we have a set of boolean attributes that can change SVM coefficients.

Therefore, the appended dimensions by $\mathcal{DN}$ can be used by each SVM in the ensemble. The randomness introduced makes these dimensions be different for each SVM, so diverse hyperplanes are obtained each time.

3 Results

Validation is made implementing Disturbing Neighbors in Java within WEKA [7]. We tested our method using WEKA ensemble implementations. Default WEKA parameters were used unless otherwise indicated. We compared our method with:

1. Bagging [1].
2. Boosting: We used AdaBoost [3] and MultiBoost [8]. In both Boosting versions we considered resampling and reweighting variants, which are respectively denoted as (S) or (W) in tables. In reweighting variant all the instances from the training dataset are used by each base classifier, but in each new round Boosting changes the weight distribution to focus into hard to classify instances. In resampling variant base classifiers are trained only with a sample of the training set according to such Boosting weight distribution.
3. Random Subspaces [2]: We tested two configurations, picking 50% and 75% of the original problem dimensions.

The size of the ensemble was fifty in all the experiments. The base method used for testing is WEKA implementation of SMO (Sequential Minimal Optimization) [9]. Linear Kernel was used. For ensembles using Disturbing Neighbors our $\mathcal{DN}$-SVM implementation is used as base classifier, setting the number of neighbors $m = 10$.

We also included in the study an ensemble with fifty $\mathcal{DN}$-SVM base classifiers to check if they perform well by their own without any sophisticated combination schema, just a simple average of the predictions generated by the individual base classifiers. We denote this method by $\mathcal{DN}$-Ensemble from now on.

Finally, we wanted to know:

1. If 1-NN accuracy was strong enough to be the main reason the disturbed classifiers could improve ensembles accuracy. So, we included this method to the test.

2. If $\mathcal{DN}$-SVM accuracy is significantly better than SVM accuracy. We wanted to test if improvement on ensembles comes from diversity enhancement or from base classifiers accuracy enhancement. So $\mathcal{DN}$-SVM on its own was added to the test.

NN methods are very robust with respect to slight variations of data set, so they do not improve very much when combined with standard ensembles [10]. Thus, we have not considered ensembles of k-NN in our test. In particular, Bagging using 1-NN as base classifiers is equivalent to 1-NN [11]. Moreover, Bagging can slightly degrade the performance of stable algorithms (e.g., k-NN) [12].

Table 1. Summary of the data sets used in the experiments.#N: Numeric features, #D: Discrete features, #E: Examples, #C: Classes. A bullet represents the $\mathcal{DN}$-version of an ensemble wins the corresponding plain version in such dataset. Columns are labeled as follows : (1) $\mathcal{DN}$-Bagging vs. Bagging, (2) $\mathcal{DN}$-MultiBoost(S) vs. MultiBoost(S), (3) $\mathcal{DN}$-MultiBoost(W) vs. MultiBoost(W), (4) $\mathcal{DN}$-AdaBoost(S) vs. AdaBoost(S), (5) $\mathcal{DN}$-AdaBoost(W) vs. AdaBoost(W), (6) DN-Subspaces (75%) vs. Subspaces (75%), (7) DN-Subspaces (50%) vs. Subspaces (50%).

Dataset	#N	#D	#E	#C	1	2	3	4	5	6	7	Dataset	#N	#D	#E	#C	1	2	3	4	5	6	7
abalone	7	1	4177	28	•						•	lymphography	3	15	148	4				•			
anneal	6	32	898	6	•			•	•	•	•	mushroom	0	22	8124	2							•
audiology	0	69	226	24			•	•		•		nursery	0	8	12960	5	•	•	•	•	•	•	•
autos	15	10	205	6	•	•	•	•	•	•	•	optdigits	64	0	5620	10	•	•	•	•	•		•
balance-scale	4	0	625	3	•	•	•	•	•	•	•	page	10	0	5473	5	•	•	•	•	•	•	•
breast-w	9	0	699	2	•				•	•	•	pendigits	16	0	10992	10	•	•	•	•	•	•	•
breast-y	0	9	286	2	•	•	•			•	•	phoneme	5	0	5404	2	•	•	•	•	•	•	•
bupa	6	0	345	2	•	•	•	•	•	•	•	pima	8	0	768	2							•
car	0	6	1728	4	•	•	•	•	•	•	•	primary	0	17	339	22	•		•		•		•
credit-a	6	9	690	2	•	•		•	•	•	•	promoters	0	57	106	2				•			
credit-g	7	13	1000	2		•	•				•	ringnorm	20	0	300	2	•	•	•	•	•	•	•
crx	6	9	690	2	•		•			•	•	sat	36	0	6435	6	•	•	•	•	•	•	•
dna	0	180	3186	3	•	•	•	•	•		•	segment	19	0	2310	7	•	•	•	•	•	•	•
ecoli	7	0	336	8	•	•	•	•		•	•	shuttle	9	0	58000	7	•	•	•	•	•	•	•
glass	9	0	214	6	•	•	•	•	•	•	•	sick	7	22	3772	2	•	•	•	•	•	•	
heart-c	6	7	303	2	•	•						sonar	60	0	208	2	•	•	•	•	•	•	•
heart-h	6	7	294	2	•					•	•	soybean	0	35	683	19	•	•	•	•	•		
heart-s	5	8	123	2		•	•	•	•	•		soybean-small	0	35	47	4		•		•			
heart-statlog	13	0	270	2	•						•	splice	0	60	3190	3	•	•		•	•		•
heart-v	5	8	200	2	•	•			•			threenorm	20	0	300	2	•	•	•	•	•		•
hepatitis	6	13	155	2		•		•	•			tic-tac-toe	0	9	958	2		•	•	•	•	•	•
horse-colic	7	15	368	2	•	•	•	•	•	•	•	twonorm	20	0	300	2	•	•	•	•	•		
hypo	7	18	3163	2	•	•	•	•	•	•	•	vehicle	18	0	846	4	•	•	•	•	•	•	
ionosphere	34	0	351	2	•	•	•	•	•	•	•	vote1	0	15	435	2		•		•			
iris	4	0	150	3		•						voting	0	16	435	2		•			•	•	
krk	6	0	28056	18	•	•	•	•	•	•	•	vowel-context	10	2	990	11	•	•	•	•	•	•	•
kr-vs-kp	0	36	3196	2	•	•	•	•	•	•		vowel-nocontext	10	0	990	11	•	•	•	•	•	•	•
labor	8	8	57	2	•	•		•			•	waveform	40	0	5000	3	•		•				
led-24	0	24	5000	10						•	•	yeast	8	0	1484	10	•					•	•
letter	16	0	20000	26	•	•	•		•	•	•	zip	256	0	9298	10	•	•	•	•	•		•
lrd	93	0	531	10					•			zoo	1	15	101	7	•						•

Table 2. Ensemble methods sorted by their average rank

Position	Average Rank	Method	Position	Average Rank	Method
1	6.31	$\mathcal{DN}$-Bagging	10	9.65	MultiBoost (W)
2	6.70	$\mathcal{DN}$-MultiBoost (S)	11	9.69	MultiBoost (S)
3	6.93	$\mathcal{DN}$-Subspaces (75%)	12	9.80	$\mathcal{DN}$-SVM
4	7.56	$\mathcal{DN}$-Ensemble	13	10.23	Subspaces (75%)
5	7.88	$\mathcal{DN}$-MultiBoost (W)	14	10.55	SVM
6	8.58	Bagging	15	11.25	AdaBoost (W)
7	9.52	$\mathcal{DN}$-AdaBoost (S)	16	12.12	1-Nearest Neighbor
8	9.61	$\mathcal{DN}$-AdaBoost (W)	17	12.38	Subspaces (50%)
9	9.61	$\mathcal{DN}$-Subspaces (50%)	18	12.64	AdaBoost (S)

Table 3. Wins, ties and losses of $\mathcal{DN}$ methods. Table a shows comparison of methods with and without $\mathcal{DN}$ based diversity. Table b shows comparison of ensemble methods with the 1-Nearest Neighbor classifier.

Method	Win-Tie-Loss
Bagging	47-5-10
Subspaces (50%)	44-4-14
Subspaces (75%)	37-2-23
AdaBoost (W)	41-2-19
AdaBoost (S)	42-1-19
MultiBoost (W)	39-4-19
MultiBoost (S)	45-2-15
SVM	31-1-30

a.

Method	Win-Tie-Loss
$\mathcal{DN}$-Ensemble	39-4-19
$\mathcal{DN}$-Bagging	40-3-19
$\mathcal{DN}$-Subspaces (50%)	33-1-28
$\mathcal{DN}$-Subspaces (75%)	45-3-14
$\mathcal{DN}$-AdaBoost (W)	42-2-18
$\mathcal{DN}$-AdaBoost (S)	40-1-21
$\mathcal{DN}$-MultiBoost (W)	41-2-19
$\mathcal{DN}$-MultiBoost (S)	44-1-17
$\mathcal{DN}$-SVM	41-1-20

b.

For validation we used the 62 UCI datasets [13] in Table 1 and 5×2 stratified cross validation, which provides an acceptable number of repetitions [14]. Results are summarized in Tables 1, 2 and 3.

Table 2 shows the methods using the average ranks from [15]. A number is assigned to each method and dataset corresponding to its rank position in such dataset. If there are ties, average ranks are assigned. Then, for each method, the average position is calculated over all datasets (see second column of Table 2). The methods are then ordered using these values. We can see that all undisturbed ensemble methods were improved by their $\mathcal{DN}$ version. $\mathcal{DN}$-Ensemble is in fourth place, thus a very competitive ensemble can be built just by using $\mathcal{DN}$-SVM as base learner.

The improvement of using $\mathcal{DN}$ versions is quantified in Table 3.a that shows wins, ties and loses of disturbed ensemble versions against undisturbed versions. Bullets in Table 1 shows datasets where $\mathcal{DN}$ versions win the plain ones. Fig. 2 points out accuracy enhancements for Bagging (the best ranked plain method).

According to the sign test [15], for 62 data sets, one method is better than other with significance level of 5%, if the number of wins plus half the ties is greater or equal than 39. Hence, for all the methods in Table 3.a, except Random Subspaces 75% and SVM,

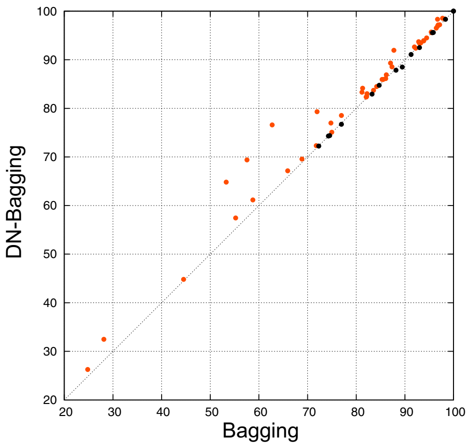


Fig. 2. Bagging vs. $\mathcal{DN}$-Bagging. Accuracy of each method is represented in each axis. Light points above the diagonal represent datasets in which $\mathcal{DN}$-Bagging is better than Bagging.

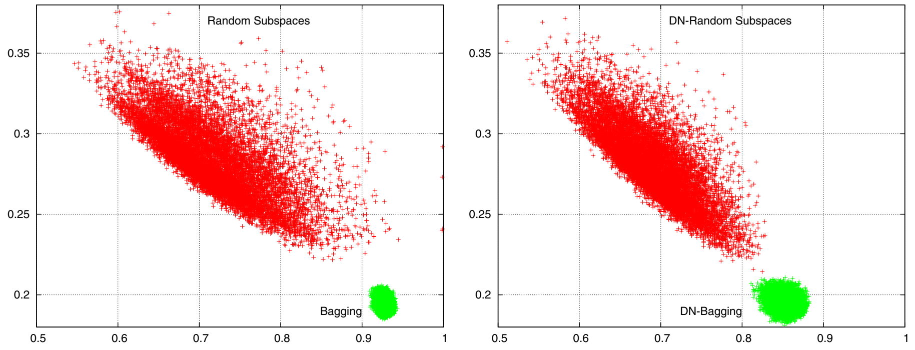


Fig. 3. Error vs. Kappa for Bagging and Random Subspaces 75% in *letter* dataset

the $\mathcal{DN}$ version is significantly better. The difference in Random Subspaces 75% is 38, so even that method is very near to show a significant improvement. $\mathcal{DN}$-SVM wins are almost the same than losses. Thus $\mathcal{DN}$-SVM does not seem better or worse than SVM. Table 3.b shows that all $\mathcal{DN}$ methods but Random Subspaces 50% are significantly better than 1-NN. So it seems that $\mathcal{DN}$ versions are not improved bacause of its 1-NN classifier. The reason of enhancement in ensembles using $\mathcal{DN}$ could be that $\mathcal{DN}$-SVM base classifiers are more diverse than SVM base classifiers without getting an important accuracy loss.

We also tested diversity improvement of $\mathcal{DN}$-SVM using the Kappa statistic [16]. Kappa measures how diverse two classifiers are, it can take values ranged from -1 to 1. A Kappa value equal to 1 means that both classifiers agree in every example, a value equal to 0 means that there is no agreement above that expected by chance,

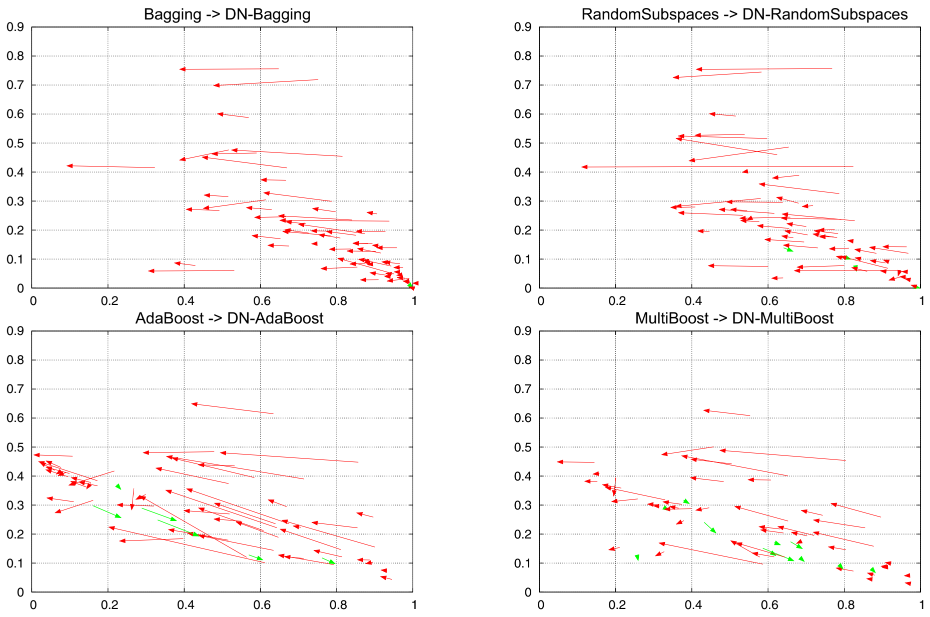


Fig. 4. Kappa-Error Movement Diagrams for the considered methods

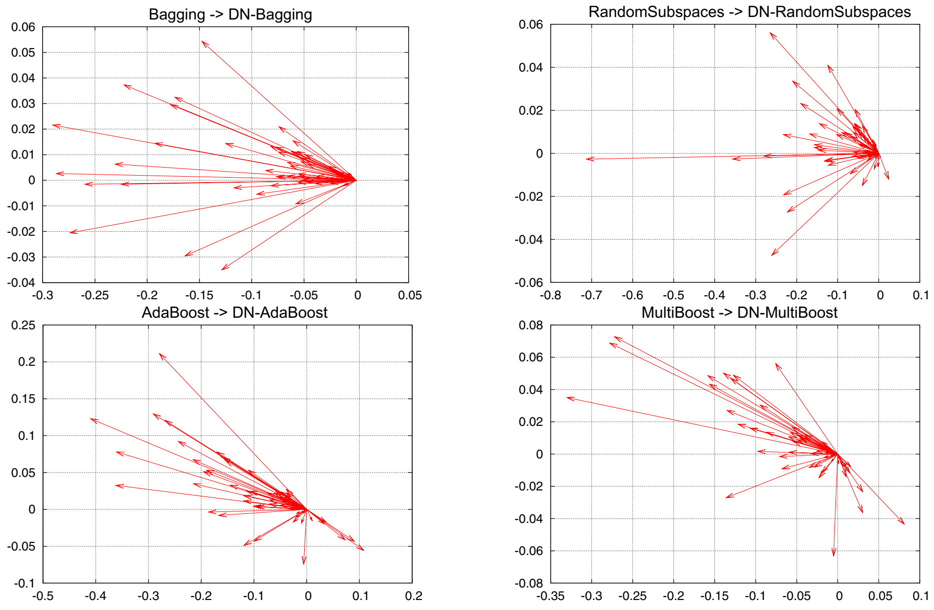


Fig. 5. Kappa-Error Relative Movement Diagrams for the considered methods

and negative Kappa values happen when there is disagreement between the classifiers. Then Kappa values are used to draw Kappa-Error Diagrams [16]. Fig. 3 shows an example with the Bagging and Random Subspaces 75% methods and the UCI dataset *letter*. We plot a point (x, y) for each pair of base classifiers belonging to the same ensemble, where x is kappa measure for these two classifiers, and y is their average error. So, ideally, pairs of base classifiers will be close to left bottom corner, because it means they are accurate and diverse. In Fig. 3 we see $\mathcal{DN}$-clouds slightly displaced to the left of undisturbed ensembles clouds. It means that $\mathcal{DN}$-methods are more diverse.

In Fig. 4 each dataset and method cloud have been substituted by the average of their points. Then, arrows are drawn from the points representing the average result of each undisturbed method and dataset to the points of the corresponding disturbed method and dataset (e.g., from letter tested with Bagging to letter tested with $\mathcal{DN}$-Bagging). Finally, these arrows are gathered into four *Kappa-Error Movement Diagrams*, one for each considered method (i.e., Bagging, Random Subspaces 75%, and both Boosting variants with resampling). We can see almost all arrows pointing left which indicates an increase in diversity, the longer the arrow the bigger the increase in diversity.

Finally, Fig. 5 shows *Kappa-Error Relative Movement Diagrams* for each ensemble method. These diagrams are obtained gathering all arrows in Fig. 4 for an ensemble, and translating the starting point of every arrow to the origin of coordinates. The majority of arrows point to left, which is an indicator of diversity growth. Many arrows also point up showing that generally, increase of diversity is at the expense of individual base classifiers accuracy.

4 Conclusion

Disturbing Neighbors is a method for altering normal training process of base classifiers in an ensemble, enhancing its diversity and improving the ensemble overall accuracy. Disturbing Neighbors builds new features using an 1-NN classifier. These features are the 1-NN output plus a set of boolean attributes indicating which is the nearest neighbor. The 1-NN classifier is built using a small subset of training instances randomly selected from the original dataset. Dimensions used to compute the Euclidean distance are also selected randomly. These two sources of randomness are the reasons why the built features are different each time, so when these new features are used to train base classifiers the diversity is increased.

Kappa statistic and Kappa movement diagrams show that $\mathcal{DN}$ supplies extra diversity to SVM base learners. Experimental validation also shows that $\mathcal{DN}$-SVM itself does not significantly improves SVM accuracy, whereas SVM ensembles that use $\mathcal{DN}$ are usually significantly better than the versions without $\mathcal{DN}$. So this improvement can only come from the increment in diversity obtained by applying $\mathcal{DN}$.

Acknowledgments. This work has been supported by the “Junta de Castilla y León” project BU007B08.

References

1. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
2. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
3. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Thirteenth International Conference on Machine Learning, pp. 148–156. Morgan Kaufmann, San Francisco (1996)
4. Vapnik, V.N.: The Nature of Statistical Learning Theory (Information Science and Statistics). Springer, Heidelberg (1999)
5. Lin, C.: Liblinear (2008), `http://mloss.org/software/view/61/`
6. Maudes, J., Rodríguez, J.J., García-Osorio, C.: Disturbing neighbors diversity for decision forests. In: Okun, O., Valentini, G. (eds.) Workshop on Supervised and Unsupervised Ensemble Methods and their Applications, SUEMA 2008, pp. 67–71 (2008)
7. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005), `http://www.cs.waikato.ac.nz/ml/weka`
8. Webb, G.I.: Multiboosting: A technique for combining boosting and wagging. Machine Learning 40(2) (2000)
9. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Advances in kernel methods: support vector learning, pp. 185–208. MIT Press, Cambridge (1999)
10. Domeniconi, C., Yan, B.: Nearest neighbor ensemble. In: ICPR, vol. (1), pp. 228–231 (2004)
11. Caprile, B., Merler, S., Furlanello, C., Jurman, G.: Exact bagging with k-nearest neighbour classifiers. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 72–81. Springer, Heidelberg (2004)
12. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36(1-2), 105–139 (1999)
13. Asuncion, A., Newman, D.: UCI machine learning repository (2007), `http://www.ics.uci.edu/~MLearn/MLRepository.html`
14. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation 10(7), 1895–1923 (1998)
15. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
16. Margineantu, D.D., Dietterich, T.G.: Pruning adaptive boosting. In: Proc. 14th International Conference on Machine Learning, pp. 211–218. Morgan Kaufmann, San Francisco (1997)

A Labelled Graph Based Multiple Classifier System*

Wan-Jui Lee and Robert P.W. Duin

Faculty of Electrical Engineering, Mathematics and Computer Sciences,
Delft University of Technology, The Netherlands
W.J.Lee@tudelft.nl, r.duin@ieee.org

Abstract. In general, classifying graphs with labelled nodes (also known as labelled graphs) is a more difficult task than classifying graphs with unlabelled nodes. In this work, we decompose the labelled graphs into unlabelled subgraphs with respect to the labels, and describe these decomposed subgraphs with the travelling matrices. By utilizing the travelling matrices to calculate the dissimilarity for all pairs of subgraphs with the JoEig approach[6], we can build a base classifier in the dissimilarity space for each label. By combining these label base classifiers with the global structure base classifiers built on dissimilarities of graphs considering the full adjacency matrices and the full travelling matrices, respectively, we can solve the labelled graph classification problem with the multiple classifier system.

1 Introduction

Multiple classifier system [5] which is an efficient technique for improving the classification performance grows rapidly in the field of statistical pattern recognition in the last decade. But strikingly, there are very few attempts [1,10] in the literature to create base classifiers in the structural pattern recognition domain [2]. In structural pattern recognition, graphs are a general and powerful data structure for object representation. The nodes in a graph can represent different objects and the relationships between these objects or parts of the objects are represented by edges. Also, labels and attributes for the nodes and edges can further be used to incorporate more information in a graph representation.

One of the few examples for creating structural base classifiers is discussed in [12]. The idea is to generate different graph-based classifiers by randomly removing nodes and their incident edges from the training graphs until a maximum number of nodes is reached for all graphs. Because of the randomness, different graph-based classifiers can be created and each becomes a base classifier in the multiple classifier system. However, with this setting, we still need to compute similarity/dissimilarity for labelled graphs using time-consuming techniques such as the maximum common subgraph [2] or the graph edit distance [7] considering a labelled graph classification problem.

* We acknowledge financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250).

Unlike graphs with unlabelled nodes, graphs with labelled nodes usually need to be processed and described with more complicated algorithms and structures. Also, classifying graphs with labelled nodes is a more difficult task than classifying graphs with unlabelled nodes. In this work, we propose a method to decompose labelled graphs into sets of unlabelled subgraphs, and describe the decomposed subgraphs with the travelling matrices. By using these travelling matrices as the adjacency matrices, we can reduce the problem of classifying labelled graphs into classifying sets of unlabelled graphs.

For each label, we can find out its corresponding nodes in a graph and calculate the travelling distances between all pairs of these nodes using Dijkstra's algorithm [3] for finding the shortest path given a pair of nodes. With the travelling distances, the connectivity information within the subgraph constructed by these corresponding nodes can be described with the travelling matrix. In the travelling matrix, the diagonal elements are always zero (a node is unreachable with itself) and the rest of the elements are the inverse of the travelling distances between nodes. Note that for a fully connected graph, the travelling matrix will reduce to an adjacency matrix. As a result, for a certain label, the travelling matrix of the subgraph for each graph can be found and used to represent the local structure. With this local representation, we can compute the dissimilarity between subgraphs with graph comparison methods. In this work, we adopt the JoEig (Joint Eigenspace) [6] approach to calculate the dissimilarity between pairs of subgraphs. The JoEig is an eigendecomposition based approach for comparing graphs. The main idea is to project a pair of graphs into a joint eigenspace which is expanded by the eigenvectors of both graphs and to compare the projected graphs. After the dissimilarity between all pairs of subgraphs are derived, we can create a base classifier in the dissimilarity space [8] for this label.

However, these label base classifiers only consider local structures of graphs. Obviously, there are also needs for base classifiers considering global structures of graph. Therefore, we also consider base classifiers with two different global structures of graphs. One is with the full adjacency matrix and the other is with the full travelling matrix. By combining the label base classifiers and the global structure base classifiers, we can solve the labelled graph classification problem with unlabelled graph representations.

The rest of the paper is organized as follows. In Section 2, we recap the JoEig approach for comparing unlabelled graphs. A multiple classifier system utilizes the label information of graphs is proposed in Section 3. Simulation results are presented in Section 4. Finally, a conclusion is given in Section 5.

2 Unlabelled Graph Comparison

Before we introduce the JoEig [6] approach for unlabelled graph comparison, some definitions and introduction on graphs are given as in the following.

A graph is a set of nodes connected by edges in its most general form. Consider the undirected graph $G = (V, E, W)$ with the node set $V = \{v_1, v_2, \ldots, v_n\}$, the edge set $E = \{e_1, e_2, \ldots, e_m\} \subset V \times V$, and the weight function $W : E \rightarrow (0, 1]$.

If the graph edges are weighted, the adjacency matrix A for the graph G is the $n \times n$ matrix with elements

$$A_{ij} = \begin{cases} W(v_i, v_j), & \text{if } (v_i, v_j) \in E; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Clearly since the graph is undirected, the matrix A is symmetric. The Laplacian [9] of the graph is defined by $L = D - A$, where D is the diagonal node degree matrix whose elements $D_{ii} = \sum_{k=1}^{n} A_{ik}$. The Laplacian matrix of G is positive semidefinite and singular, and it is more often adopted for spectral analysis than the adjacency matrix because of its properties.

2.1 JoEig: Graph Comparison in Joint Eigenspace

JoEig projects each pair of two graphs into a joint eigenspace. This joint eigenspace is expanded by both set of eigenvectors.

Let G and H be weighted undirected graphs and L_G and L_H be their Laplacian matrices, respectively. The eigendecomposition of L_G and L_H are performed as

$$L_G = V_G D_G V_G^T, \ L_H = V_H D_H V_H^T, \tag{2}$$

where V_G and V_H are orthonormal matrices and D_G and D_H are diagonal matrices of the eigenvalues (in ascending order) of G and H, respectively. With the joint projection vector $V_G V_H^T$, both graphs G and H will be projected to their joint eigenspace as $L_G V_G V_H^T$ and $V_G V_H^T L_H$. The difference between two graphs using JoEig is defined as

$$\|V_G D_G V_H^T - V_G D_H V_H^T\|^2. \tag{3}$$

The JoEig approach approximates a graph by relocating its eigenvalues in the joint eigenspace constructed by the eigenvectors of both graphs.

There are also three possibilities for setting the number of eigenvectors to compare graphs with different sizes in JoEig. In this work, we choose to make full use of the eigenvectors from the smaller graph and keep the same number of eigenvectors and eigenvalues in the larger graph as in the smaller graph by removing less important eigenvalues and eigenvectors from the larger graph.

3 A Labelled Graph Based Multiple Classifier System

We use the example graph shown in Figure 1(a) through this section to explain our method. This example graph is with 8 nodes and each node is labelled with one symbol. There are no attributes on the edges and the elements of the adjacency matrix A given in Eq.(4) of this graph are either 1 or 0 to indicate whether there is an edge between two nodes or not.

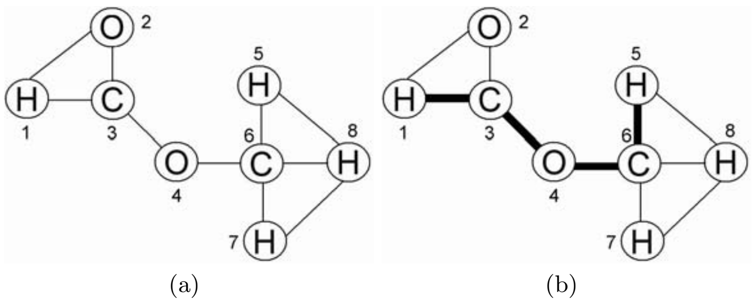


Fig. 1. An example of (a) labelled graph; (b) the shortest path between node 1 and node 5

$$A = \begin{pmatrix} 0&1&1&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0 \\ 1&1&0&1&0&0&0&0 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&0&0&1&0&1 \\ 0&0&0&1&1&0&1&1 \\ 0&0&0&0&0&1&0&1 \\ 0&0&0&0&1&1&1&0 \end{pmatrix} . \tag{4}$$

3.1 Travelling Matrix

Our goal is to solve the labelled graph classification problem by decomposing labelled graphs into sets of unlabelled subgraphs. In order to represent the decomposed subgraphs, we need an other way than by the adjacency matrix to describe the connectivity information between nodes. The main reason is that, if we only choose nodes with the same label to form a subgraph, there are probably nodes with no neighbors at all and the subgraph might fall into isolated parts if it is represented by the adjacency matrix. To avoid this phenomenon, we propose the travelling matrix to represent the connectivity information of subgraphs. The basic assumption is that the larger the travelling distance between two nodes is, the less connective they are. The travelling distance between a pair of nodes can be easily computed with Dijkstra's shortest path algorithm [3]. An example of the shortest path between node 1 and node 5 is given in Figure 1(b), and the travelling distance between these two nodes is 4 since there are at least 4 edges one node has to travel to reach the other one. Therefore, an element in the travelling matrix is defined as the inverse of the travelling distance between two nodes. Also, the elements on the diagonal are all defined as zero. For the example graph in Figure 1(a), its full travelling matrix T will be

$$T = \begin{pmatrix} 0 & 1 & 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{3} & \frac{1}{4} & \frac{1}{4} \\ 1 & 0 & 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{3} & \frac{1}{4} & \frac{1}{4} \\ 1 & 1 & 0 & 1 & \frac{1}{3} & \frac{1}{2} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 1 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 & 1 & \frac{1}{2} & 1 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{2} & 1 & 1 & 0 & 1 & 1 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 1 & 0 & 1 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 1 & 1 & 1 & 0 \end{pmatrix} . \qquad (5)$$

Note that for a fully connected graph, the travelling matrix will reduce to an adjacency matrix. For the example in Figure 1(a), there are three different labels, i.e., C, H and O. For each label, we will extract a subgraph consisting only nodes with this particular label. For example, Figure 2(a), Figure 2(b) and Figure 2(c) are the subgraphs extracted with label H, O, and C, respectively. The solid line means that these two nodes are connected in the original graph, and the dash line means they are not connected in the original graph but now weakly connected by their travelling information. Now that a subgraph only consists of nodes with the same label, it means that we can actually ignore the label within the subgraph and fully describe this subgraph with a connectivity matrix (which is, travelling matrix by our definition). The travelling matrices for these 3 subgraphs are

$$T_H = \begin{pmatrix} 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{2} & 1 \\ \frac{1}{4} & \frac{1}{2} & 0 & 1 \\ \frac{1}{4} & 1 & 1 & 0 \end{pmatrix}, T_O = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}, \text{ and } T_C = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}, \text{ respectively.}$$

3.2 Dissimilarity and Base Classifiers

Given m graphs with n distinctive labels among the graphs, we want to create n base classifiers with respect to the labels. So, for a certain label, we extract a subgraph and its travelling matrix from each graph consisting only with this label

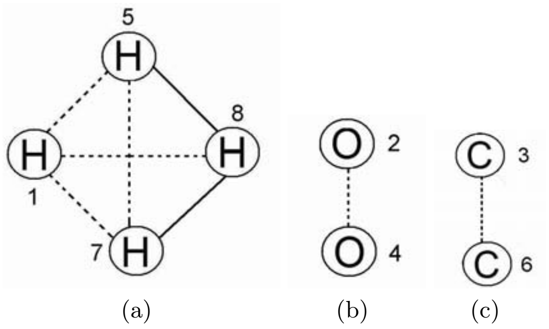


Fig. 2. Examples of subgraphs extracted with label (a) H, (b) O and (c) C, respectively, from the graph in Figure 1(a)

as described above. With these m subgraphs, the dissimilarity are calculated pairwise with the JoEig approach as described in Section 2. Some graphs might have no nodes with such label at all, and therefore the subgraphs of these graphs are empty. In this case, we directly set the dissimilarity to 0 if the two subgraphs are both empty, and set the dissimilarity to 1 if only one of the subgraphs is empty. As a result, we can obtain a $m \times m$ dissimilarity matrix for each label. With this dissimilarity matrix, we can build a base classifier for this label in the dissimilarity space [8]. In the end, we can construct n label base classifiers by doing the same to each label.

However, the label base classifiers only consider subgraphs which describe the local structures of graphs. To increase the diversity of the multiple classifier system, we also create global structure base classifiers. We propose two different global structure base classifiers, one for the full travelling matrix and the other for the full adjacency matrix. So we pairwise compare the original graphs with the JoEig approach to derive the dissimilarity matrix. But these original graphs can be represented with the full travelling matrices or the full adjacency matrices. Similar to the above, we also build global structural base classifiers for these two dissimilarity matrices.

4 Experiments

In this section, we compare the performance of the single base classifiers as described in Section 3 with the classifier combiner. Two classifiers, i.e., linear discriminant (ldc) and nearest mean classifier (nmc), are adopted to build base classifiers in the dissimilarity space [8]. All the base classifiers and the classifier combiner are built with the PRTOOLS [4]. Two real-world datasets, i.e., Mutagenicity and AIDS [11], are used in the experiments. We use 15% of training objects as the representative objects to construct the dissimilarity space for both datasets. Also, the eigenvalue diagonal and eigenvector matrices are resized to the size of the smaller graph with the JoEig approach. Moreover, all the results in the following are the average over 50 repetitions of experiments resulting in a very small standard deviation.

4.1 Experiment 1: Mutagenicity Dataset

Mutagenicity is one of the numerous adverse properties of a compound that hampers its potential to become a marketable drug. The molecules are converted into graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the corresponding chemical symbol, and there are 10 different symbols in total. The average number of nodes of a graph is 30.3177 ± 20.1201, and the average number of edges is 30.7694 ± 16.8220. The Mutagenicity dataset is divided into two classes, i.e., mutagen and nonmutagen. There are in total 4,337 elements (2,401 mutagen elements and 1,936 nonmutagen elements). In the experiments, 40% of objects are randomly selected as the training dataset, 30% are taken as the validation set and the other 30% are used as the testing dataset.

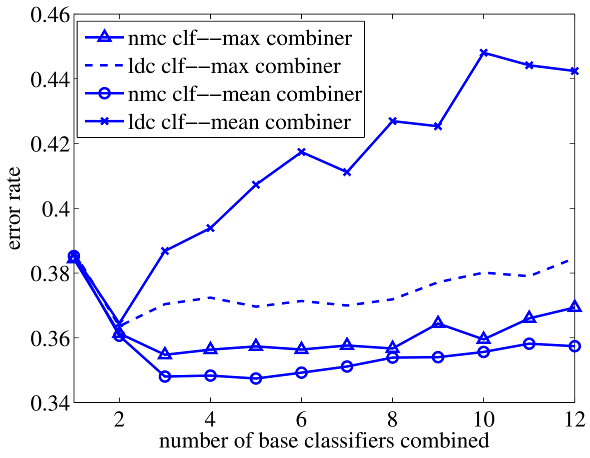


Fig. 3. Combination results of different number of base classifiers for Mutagenicity dataset

In Figure 3, we add the base classifiers (10 label base classifiers and 2 global structure classifiers) one by one with the sequential forward feature selection technique. The base classifier contributes most (with respect to the validation dataset) to the combination results of the current chosen base classifiers will be selected as the next base classifier to be combined. From Figure 3, we can see that nmc base classifiers give better combination results than ldc base classifiers with both max and mean combining rules. Combining ldc base classifiers with the mean combining rule performs much worse than the other combinations, especially when more and more base classifiers are combined. This is because some dissimilarity matrices in the label base classifier are highly correlated as some labels are absent in most graphs. As a result, most elements in the dissimilarity matrix are zero. Therefore, these base classifiers become very noisy to ldc with the mean combining rule. Also, for nmc base classifiers, the error rates of the combination results first decrease when more classifiers are included in the combination, and then remain stable, but increase again in the end when too many worse base classifiers are included in the combination. A very interesting phenomenon is that all the combiners reaching the lowest error rate have at least one of the global structure base classifiers as the base classifiers. So it is clear that the global structures can improve the classification performance.

Now the question is, would the label base classifiers also improve the performance of the combiner or is it sufficient to only combine the global structure base classifiers? In Figure 4(a) and Figure 4(b), we present the learning curve of nmc base classifiers and their max and mean combiners, respectively. From both figures, we observe that the results of combining only two global structural classifiers are much worse than combining the best 4 base classifiers. Therefore, label base classifiers also contribute significantly to the combiner.

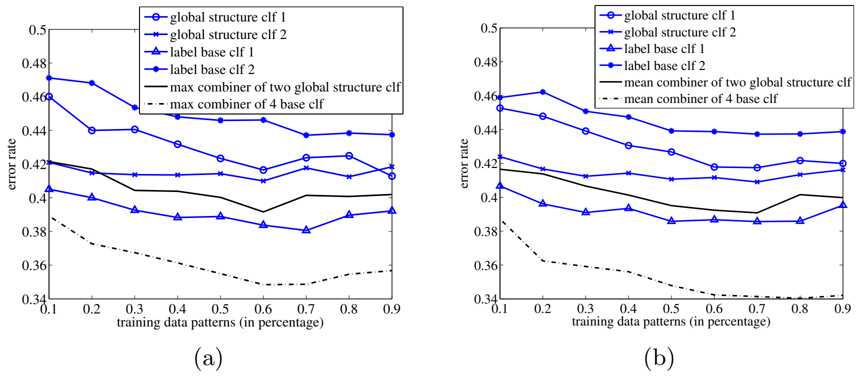


Fig. 4. Learning curves of nmc base classifiers and their (a) max and (b) mean combiners

4.2 Experiment 2: AIDS Dataset

The AIDS dataset consists of graphs representing molecular compounds. The graphs are constructed from the AIDS Antiviral Screen Database of Active Compounds (molecules). This dataset consists of two classes, active and inactive, to indicate molecules with activity against HIV or not. The molecules are converted into graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the corresponding chemical symbol, and there are 26 labels in total. The average number of nodes of a graph is 15.6953 ± 13.1918, and the average number of edges is 16.1986 ± 15.0123. There are 2,000 elements in total (1,600 inactive elements and 400 active elements). In

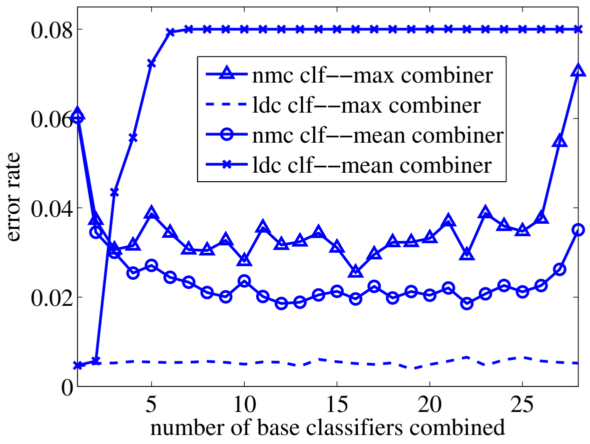


Fig. 5. Combination results of different number of base classifiers for AIDS dataset

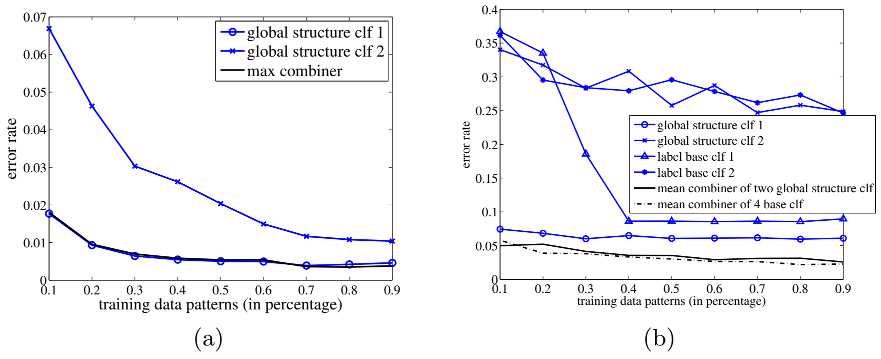


Fig. 6. Learning curves of (a) ldc base classifiers and their max combiner and (b) nmc base classifiers and their mean combiner

the experiments, 40% of objects are randomly selected as the training dataset, 30% are taken as the validation set and the other 30% are used as the testing dataset.

In Figure 5, the base classifiers (26 label base classifiers and 2 global structure classifiers) are added one by one using the same technique described above. The AIDS dataset is a much easier dataset to classify compared to the Mutagenicity dataset, and some base classifiers already reach very small error rates which makes it for the combiner difficult to improve the individual performance. From Figure 5, we can still observe that the ldc base classifiers with the mean combining rules are heavily disturbed by the correlated dissimilarity matrices and perform much worse than the other combinations. On the other hand, ldc base classifiers with the max combining perform much better than the others.

In Figure 6(a), the combiner is only slightly better than individual ldc classifiers with large amount of training data because one of the individual classifier has almost zero error rate and that leaves no much room for the combiner to improve. On the other hand, if we use weak base classifiers as in Figure 6(b), the combiners can have more significant improvements.

5 Discussions and Conclusions

We solve the labelled graph classification problem with the multiple classifier system by decomposing labelled graphs into unlabelled subgraphs with their labels and building label base classifiers from these subgraphs. Two global structural base classifiers are also considered to increase the diversity of the multiple classifier system. The subgraphs are represented by the travelling matrices instead of the adjacency matrices. The travelling matrix records the node to node travelling information. By comparing graphs/subgraphs pairwise with the JoEig approach, we can derive the dissimilarity matrix. With the dissimilarity matrix, we can construct the base classifier in the dissimilarity space.

Even though we only consider nodes with single labels and edges with no attributes in the experiments, our approach also applies to nodes with multiple labels and edges with attributes. For multiple labels, we can decompose graphs into subgraphs that might have common nodes. For attributed edges, we can simply use the weighted adjacency matrix.

References

1. Bunke, H., Irniger, C., Neuhaus, M.: Graph Matching - Challenges and Potential Solutions. In: Roli, F., Vitulano, S. (eds.) ICIAP 2005. LNCS, vol. 3617, pp. 1–10. Springer, Heidelberg (2005)
2. Bunke, H., Riesen, K.: Graph Classification Based on Dissimilarity Space Embedding. In: da Vitoria, L., et al. (eds.) Proc. SSSPR 2008, Structural, Syntacic, and Statistical Pattern Recognition, Florida, USA. LNCS, vol. 5342, pp. 996–1007. Springer, Heidelberg (2008)
3. Dijkstra, E.W.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1, 269–271 (1959)
4. Duin, R.P.W., Juszczak, P., Paclik, P., Pękalska, E., de Ridder, D., Tax, D.M.J.: PRTOOLS4, A Matlab Toolbox for Pattern Recognition, The Netherlands, Delft University of Technology. ICT Group (2004), http://www.prtools.org
5. Kuncheva, L.I.: Combining Pattern Classifiers. Methods and Algorithms. Wiley, Chichester (2004)
6. Lee, W.J., Duin, R.P.W.: An Inexact Graph Comparison Approach in Joint Eigenspace. In: Proc. SSSPR 2008, Structural, Syntacic, and Statistical Pattern Recognition, Florida, USA. LNCS, vol. 5342, pp. 35–44. Springer, Heidelberg (2008)
7. Neuhaus, M., Bunke, H.: Edit Distance-Based Kernel Functions for Structural Pattern Classification. Pattern Recognition 39, 1852–1863 (2006)
8. Pękalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition. In: Fundations and Applications. World Scientific, Singapore (2005)
9. Qiu, H.J., Hancock, E.R.: Spectral Simplication of Graphs. In: Proceedings of the 8th European Conference on Computer Vision, Czech Republic, pp. 114–126 (2004)
10. Riesen, K., Bunke, H.: Classifier Ensembles for Vector Space Embedding of Graphs. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 220–230. Springer, Heidelberg (2007)
11. Riesen, K., Bunke, H.: IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In: da Vitoria, L., et al. (eds.) Proc. SSSPR 2008, Structural, Syntacic, and Statistical Pattern Recognition, Florida, USA. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)
12. Schenker, A., Bunke, H., Last, M., Kandel, A.: Building Graph-Based Classifier Ensembles by Random Node Selection. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 214–222. Springer, Heidelberg (2004)

Cluster Ensembles Based on Vector Space Embeddings of Graphs

Kaspar Riesen and Horst Bunke

Institute of Computer Science and Applied Mathematics, University of Bern,
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{riesen,bunke}@iam.unibe.ch

Abstract. Cluster ensembles provide us with a versatile alternative to individual clustering algorithms. In structural pattern recognition, however, cluster ensembles have been rarely studied. In the present paper a general methodology for creating structural cluster ensembles is proposed. Our representation formalism is based on graphs and includes strings and trees as special cases. The basic idea of our approach is to view the dissimilarities of an input graph g to a number of prototype graphs as a vectorial description of g. Randomized prototype selection offers a convenient possibility to generate m different vector sets out of the same graph set. Applying any available clustering algorithm to these vector sets results in a cluster ensemble with m clusterings which can then be combined with an appropriate consensus function. In several experiments conducted on different graph sets, the cluster ensemble shows superior performance over two single clustering procedures.

1 Introduction

Clustering, a common task in pattern recognition and related fields, refers to the process of dividing a set of given objects into homogeneous groups referred to as clusters. Cluster ensembles have been introduced as a more accurate alternative to individual clustering algorithms [1]. The basic idea of ensemble methods is to combine the partitions of many clustering algorithms applied to a specific data set to one final decision.

A large amount of clustering algorithms based on pattern representations in terms of feature vectors have been proposed in the literature (see [2] for a survey). Also, quite a number of papers are concerned with cluster ensemble methods based on feature vectors [3,4,5,6]. Contrariwise, there are only few works where symbolic data structures, and in particular graphs, are used for data clustering [7]. This is rather surprising since in many application domains there exist data sets that possess inherent structural or relational characteristics. Obviously, such data is highly suitable for graph based representations. Prominent examples are network topologies [8], molecular compounds [9], and web graphs [10]. We refer to [11] for an exhaustive review of graph based representation in pattern recognition.

The lack of graph clustering algorithms arises from the fact that there is little mathematical structure in the domain of graphs. For example, computing the sum, the weighted sum, or the product of a pair of entities (which are elementary operations, required in many clustering algorithms) is not possible or not defined in a standardized way in the domain of graphs. However, graph kernels, a relatively novel class of algorithms for pattern recognition, offer an elegant solution to overcome this drawback of graph based representation [12]. Originally, kernel methods have been developed for transforming a given feature space into another one of higher dimensionality without computing the transformation explicitly for each individual feature vector. Recently, however, as a fundamental extension, the existence of kernels for symbolic data structures, especially for graphs, has been shown [13].

In the present paper we address the problem of graph clustering by means of a graph kernel based on the idea of dissimilarity space embedding. The idea of such embeddings was originally developed in order to map sets of feature vectors in a dissimilarity space [14]. Later this procedure has been transferred from vectors to strings [15], and eventually to graphs [16]. The general idea of our approach is to represent the underlying graphs by means of dissimilarities to prototype graphs. The motivation for this procedure is twofold. First, we overcome the lack of clustering algorithms in the graph domain. Secondly, this embedding procedure is particularly interesting as it offers an elegant solution to the crucial question of how to build the individual clusterings of an ensemble. As will be shown in this paper, through repeated selection of different prototype graphs one can produce an arbitrary number of vectorial descriptions of the same graph set. The clusterings resulting from these vector sets are then used as ensemble members.

The present paper extends previous work on structural classifier ensemble methods [17] to the setting of multiple clustering systems. With a comparison based on four different validation indices we empirically confirm that our novel ensemble procedure results in better clusterings when compared to results achieved with a single clustering algorithm applied in the domain of graphs as well as in the embedding space. Hence, the general procedure for building structural cluster ensembles together with an experimental study of this novel approach is the main contribution of the present paper.

The remainder of this paper is organized as follows. Next, the dissimilarity space embedding graph kernel is described. In Section 3, the idea of our cluster ensemble is introduced and the base clustering algorithm employed in our experiments is described in detail. The experimental results are presented and discussed in Section 4. Finally, in Section 5, we draw some conclusions.

2 Dissimilarity Space Embedding Graph Kernel

Kernel methods have become one of the most rapidly emerging sub-fields in intelligent information processing [12]. Recently, kernel theory has been generalized to the domain of graphs [13].

Definition 1 (Graph Kernel). *Let $\mathcal{G}$ be a finite or infinite set of graphs, g_1, $g_2 \in \mathcal{G}$, and $\varphi : \mathcal{G} \to \mathcal{F}$ a function where $\mathcal{F}$ is a feature vector space endowed with a scalar product $\langle ., . \rangle$. A graph kernel function is a mapping $\kappa : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ such that $\kappa(g_1, g_2) = \langle \varphi(g_1), \varphi(g_2) \rangle$.* $\square$

According to this definition a graph kernel function takes two graphs g_1 and g_2 as arguments and returns a real number that is equal to the result achieved by first mapping the two graphs by a function φ to a feature space $\mathcal{F}$ and then computing the scalar product $\langle \varphi(g_1), \varphi(g_2) \rangle$ in $\mathcal{F}$. That is, instead of mapping graphs from $\mathcal{G}$ to $\mathcal{F}$ and computing their scalar product there, one can simply evaluate the value of the kernel function κ in the original graph domain. This procedure is commonly referred to as the *kernel trick*.

What makes kernel theory interesting is the fact that many algorithms can be kernelized, i.e. reformulated such that only pairwise scalar products rather than explicit objects are needed[1]. Obviously, by replacing the scalar product by a valid kernel function it is possible to run kernelizable algorithms in an implicitly existing feature vector space $\mathcal{F}$.

The graph kernel used in the present paper is based on a graph embedding procedure that makes use of graph edit distance. The key idea of graph edit distance is to define the dissimilarity, or distance, of graphs by the minimum amount of distortion that is needed to transform one graph into another. A standard set of distortion operations is given by *insertions*, *deletions*, and *substitutions* of nodes and edges.

Given two graphs, the source graph g_1 and the target graph g_2, in order to compute the graph edit distance of g_1 and g_2 we delete some nodes and edges from g_1, relabel (substitute) some of the remaining nodes and edges, and insert some nodes and edges in g_2, such that g_1 is finally transformed into g_2. A sequence of edit operations $e_1, \ldots, e_k$ that transform g_1 into g_2 is called an *edit path* between g_1 and g_2. In order to find the most suitable edit path out of all possible edit paths, one introduces a cost for each edit operation, measuring the strength of the corresponding operation. The idea of such cost functions is to define whether or not an edit operation represents a strong modification of the graph. Consequently, the *edit distance* of two graphs is defined by the minimum cost edit path between two graphs. The edit distance of graphs can be computed, for example, by a tree search algorithm [18] or by faster, suboptimal methods which have been proposed recently (e.g. [19]).

Assume we have available a set of graphs $\mathcal{G} = \{g_1, \ldots, g_N\}$ and a graph dissimilarity measure $d(g_i, g_j)$ (in our case the graph edit distance). After having selected a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of $n \leq N$ prototypes from $\mathcal{G}$, we compute the dissimilarity of a given graph $g \in \mathcal{G}$ to each prototype $p \in \mathcal{P}$. This leads to n dissimilarities, $d_1 = d(g, p_1), \ldots, d_n = d(g, p_n)$, which can be arranged in an n-dimensional vector $(d_1, \ldots, d_n)$. In this way we can transform any graph from $\mathcal{G}$ into a vector of real numbers.

[1] Such algorithms together with a kernel function κ are commonly termed *kernel machines*.

Definition 2 (Graph Embedding). *If $\mathcal{G} = \{g_1, \ldots, g_N\}$ is a set of graphs and $\mathcal{P} = \{p_1, \ldots, p_n\} \subseteq \mathcal{G}$ is a set of prototype graphs, the mapping $\varphi_n^{\mathcal{P}} : \mathcal{G} \rightarrow \mathbb{R}^n$ is defined as the function*

$$\varphi_n^{\mathcal{P}}(g) \mapsto (d(g, p_1), \ldots, d(g, p_n)),$$

where $d(g, p_i)$ is the graph edit distance between graph g and the i-th prototype.

Note that in practice, the prototype set $\mathcal{P}$ is often defined on a training set $\mathcal{T}$ of graphs, i.e. $\mathcal{P} \subseteq \mathcal{T} \subset \mathcal{G}$. However, the mapping $\varphi_n^{\mathcal{P}}(.)$ is not restricted to graphs from $\mathcal{T}$, but all graphs from $\mathcal{G}$ can be mapped via $\varphi_n^{\mathcal{P}}(.)$ to $\mathbb{R}^n$, of course.

Clearly, the graph embedding procedure described above provides a foundation for a novel class of graph kernels. Based on the resulting graph maps $\varphi_n^{\mathcal{P}}$, standard kernel functions for feature vectors in $\mathbb{R}^n$ can be applied, mapping the vector space embedded graphs implicitly into a higher dimensional feature space $\mathcal{F}$. An example is the RBF kernel.

$$\kappa_{RBF}(g_1, g_2) = exp\left(-\gamma ||\varphi_n^{\mathcal{P}}(g_1) - \varphi_n^{\mathcal{P}}(g_2)||^2\right) \text{ , with } \gamma > 0.$$

Obviously, in every kernel machine the scalar product can be replaced by $\kappa(g_1, g_2)$ such that these algorithms can be applied to objects originally given in terms of graphs.

3 Cluster Ensembles

3.1 General Procedure

The two major issues in the construction of a cluster ensemble are how to build the individual clustering algorithms and how to combine their decisions [1]. Various approaches for building the ensemble members have been proposed in the literature. Among these are random initialization of the clustering algorithm [3], randomly choosing the number of clusters [4], applying different types of clustering algorithms [5], or using subsets of features [6].

In the present paper the following approach is employed. Through the embedding framework introduced in the previous section, we establish not only a general framework for mapping graphs to the real vector space $\mathbb{R}^n$, but also a straightforward approach for building the ensemble members in a multiple clustering system. Regarding the graph embedding procedure, the selection of the n prototypes $\{p_1, \ldots, p_n\}$ is a critical issue since not only the prototypes themselves but also their number affect the resulting vectors. This particular characteristic of our embedding framework is utilized for building the cluster ensemble members. That is, selecting m times the prototypes randomly, and varying the respective number of prototypes n in a certain interval for each selection, results in m different prototype sets with different cardinality $\mathcal{P}_1 = \{p_{11}, \ldots, p_{1n_1}\}, \ldots, \mathcal{P}_m = \{p_{m1}, \ldots, p_{mn_m}\}$.

For each prototype set $\mathcal{P}_i$ a mapping $\varphi_{n_i}^{\mathcal{P}_i}$ is defined according to Def. 2 which map a given graph set $\mathcal{G} = \{g_1, \ldots, g_N\}$ to m different vector sets $\mathbf{X}^{(1)} =$

$\{\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_N^{(1)}\}, \ldots, \mathbf{X}^{(m)} = \{\mathbf{x}_1^{(m)}, \ldots, \mathbf{x}_N^{(m)}\}$. Note that the dimensionality of the vectors in $\mathbf{X}^{(s)}$ depend on the number n_s of selected prototypes. Clearly, applying a clustering algorithm to each of these vector sets results in m different clusterings of the underlying data.

For the combination of the individual clusterings the pairwise approach is implemented as consensus function [1,5]. That is, for each of the m clusterings an $N \times N$ coassociation matrix is built. The entries in this coassociation matrix are either set to one or zero, depending on whether or not the two corresponding elements are in the same cluster. Obviously, the normalized sum of the m coassociation matrices can be seen as similiarity matrix between the N elements. That is, pairs of elements will have a high similarity value if they often belong to the same cluster. Contrariwise, the similarity value will be low if two elements are assigned to the same cluster only a few times. In order to generate the final ensemble clustering, this similarity matrix can now be used with any clustering algorithm which directly operates on pairwise similarities. In Algorithm 1 the cluster combination procedure is described in pseudo-code.

Algorithm 1. Clustering combination

```
Input:   Data sets $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(m)}$ with $N$ elements each, number of clusters $k$
Output:  Consensus matrix $\mathbf{M}$ (Similarity matrix) and final clustering $\{C_j\}_{j=1}^k$.

1: for each data set $\mathbf{X}^{(s)}$, $s = 1, \ldots, m$ do
2:     Generate clustering $\{C_j^{(s)}\}_{j=1}^k$ with $k$ clusters $C_1^{(s)}, \ldots, C_k^{(s)}$.
3: end for
4: for each clustering $\{C_j^{(s)}\}_{j=1}^k$, $s = 1, \ldots, m$ do
5:     Form a coassociation matrix $M^{(s)} = \{m_{ij}^{(s)}\}$, of size $N \times N$, where
           $m_{ij}^{(s)} = \begin{cases} 1 & \text{if } \mathbf{x}_i^{(s)} \text{ and } \mathbf{x}_j^{(s)} \text{ are in the same cluster} \\ 0 & \text{else} \end{cases}$
6: end for
7: Form consensus matrix $\mathbf{M} = \frac{1}{m} \sum_{s=1}^m M^{(s)}$
8: Generate final clustering $\{C_j\}_{j=1}^k$ based on the consensus matrix $\mathbf{M}$
```

Next, we describe our base clustering algorithm employed for both clustering of the individual data sets $\mathbf{X}^{(s)}$ and generation of the final clustering based on the consensus matrix $\mathbf{M}$.

3.2 Kernel *k*-Means Clustering

The k-means algorithm is one of the most popular clustering algorithms in pattern recognition and related areas. This algorithm employs a squared error criterion, i.e. it finds k clusters $C_1, \ldots, C_k$ such that the objective function

$$f\left(\{C_j\}_{j=1}^k\right) = \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} d(\mathbf{x}_i, \mathbf{m}_j)$$

is minimized. In this formula, $\mathbf{m}_j$ refers to the mean of cluster C_j.

Commonly k-means algorithm starts with a random initial partition of the data and keeps reassigning the patterns to clusters based on the similarity between the pattern and the cluster centers until a certain convergence criterion is met (e.g. no reassignment of objects from one cluster to another has taken place during the last iteration). In the present paper a deterministic initialization of the algorithm is applied. The set of initial cluster centers is constructed by iteratively retrieving the median of set $\mathbf{X}$ minus the objects already selected. The median of set $\mathbf{X}$ is the object $\mathbf{x} \in \mathbf{X}$ that minimizes the sum of distances to all other objects in $\mathbf{X}$. Obviously, this procedure initializes k-means with objects situated in, or near, the center of the set $\mathbf{X}$.

A well known drawback of k-means clustering is that the individual clusters C_j need to be spherical in order to achieve satisfactory results. (This drawback directly follows from the minimization of the squared error.) However, it turns out that the k-means algorithm can be written as a kernel machine, i.e. it can be reformulated in terms of pairwise scalar products only. Hence, we can replace the scalar products $\langle ., . \rangle$ with a valid kernel function κ to represent the scalar product in an implicit feature space $\mathcal{F}$. Applying k-means clustering in the resulting feature space $\mathcal{F}$, i.e. finding spherical clusters C_j in $\mathcal{F}$, corresponds to finding (possibly) non-spherical clusters in the original vector space $\mathbb{R}^n$. Consequently, this clustering procedure is much more powerful than the conventional k-means algorithm. The resulting procedure is commonly referred to as kernel k-means clustering.

As base clustering algorithm, kernel k-means clustering is applied in conjunction with the graph kernel defined in Section 2, i.e.

$$\kappa_{RBF}(g_1, g_2) = exp\left(-\gamma||\varphi_n^{\mathcal{P}}(g_1) - \varphi_n^{\mathcal{P}}(g_2)||^2\right)$$

with $\gamma > 0$. Also for the combination of the clustering results kernel k-means clustering is employed, interpreting the entries m_{ij} of the consensus matrix $\mathbf{M}$ as kernel values κ_{ij}.

4 Experimental Results

4.1 Databases

For our experimental evaluation, four data sets with quite different characteristics are used. The data sets vary with respect to graph size, edge density, type of labels for the nodes and edges, and meaning of the underlying objects. Lacking space we give a short description of the data only. For a more thorough description we refer to [20] where the data sets are discussed in greater detail[2].

The first database used in the experiments consists of graphs representing distorted letter line drawings out of 15 classes (Letter). Next we apply the proposed method to the problem of image clustering, i.e. we use graphs representing images

[2] Note that the data sets are publicly available: `http://www.iam.unibe.ch/fki/databases/iam-graph-database`

out of two categories (*cups, cars*) from the COIL-100 database [21] (COIL). The third data set is given by graphs representing fingerprint images of the NIST-4 database [22] out of the four classes *arch*, *left*, *right*, and *whorl* (Fingerprint). Finally, the fourth set is given by the Enzyme data set. The graphs are constructed from the Protein Data Bank [23] and labeled with their corresponding enzyme class labels (*EC 1,..., EC 6*) (Enzymes).

4.2 Cluster Validation Indices

In order to measure the quality of our clusterings, we use four different validation indices, viz. DUNN [24], C [25], RAND [26], and the BIPARTITE index [27].

DUNN index measures the ratio of the minimum distance of two different clusters and the maximum diameter of a cluster. Hence, DUNN is considered to be positively-correlated such that higher values indicate higher clustering quality. In contrast with the DUNN index, the smaller the C index value is, the higher is the clustering quality. The C index measures how frequently pairs with a small distance belong to the same cluster. RAND index measures the numbers of pairs belonging to the same class and to the same cluster and the number of pairs that neither belong to the same class nor to the same cluster. Hence, RAND index measures the consistency of a given clustering, and therefore higher values indicate better clusterings. The BIPARTITE index (BP index for short) gives us the maximum possible classification accuracy of the given clustering.

Whereas the two former indices (DUNN and C) do not need any ground truth information, the latter ones (RAND and BP) are defined with respect to the class memberships of the underlying objects.

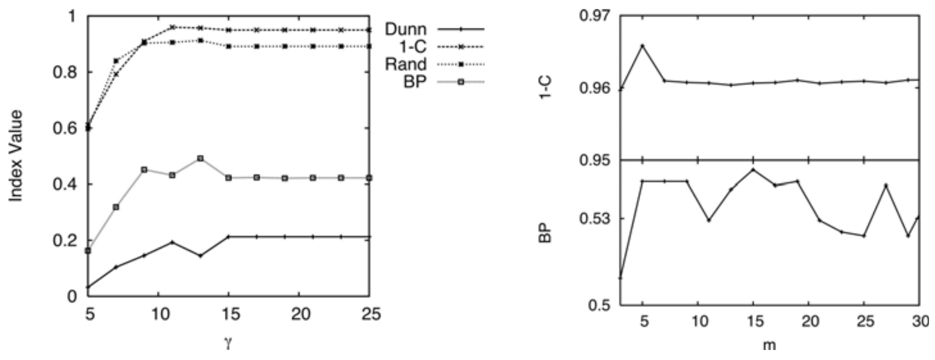
4.3 Experimental Setup

In our study on structural cluster ensembles, we use two reference systems, i.e. two single clustering algorithms. First, we apply k-means algorithm to the original graph data. Note that in this case the distance function d is given by the graph edit distance and the mean $\mathbf{m}_j$ of the j-th cluster is defined as the set median graph ($\mathbf{m}_j = argmin_{g_1 \in C_j} \sum_{g_2 \in C_j} d(g_1, g_2)$). In the remainder of the present paper we denote k-means applied to graphs as k-medians. Secondly, we use a single kernel k-means algorithm applied to the vector space embedded graphs. To this end we use the best performing ensemble member as second reference system. Note that the best performing ensemble member is defined for each validation index independently.

Each of our graph sets is divided into two disjoint subsets, viz. validation and test set. The validation set is used to determine those meta parameters of the clustering algorithm which cannot be directly inferred from the specific application. For k-medians clustering in the original graph domain only the cost function for graph edit distance has to be validated. For our novel approach, however, there are two additional parameters to tune, viz. the final ensemble size m' and the parameter γ in the RBF kernel.

First, for each vector set $\mathbf{X}^{(s)}$ $(s = 1, \ldots, m)$ the kernel value γ is optimized regarding one specific validation criterion at a time. In Fig. 1 (a) the optimization of γ is illustrated on one vector set $\mathbf{X}^{(s)}$ for all validation indices. That is, the validation criterion is plotted as a function of γ.

For the final ensemble size m' we follow the strategy which is best known as *overproduce-and-select* [28]. We first generate a large number m of clusterings optimized on the validation set (here $m = 100$). Next, starting with the best individual clustering, the ensemble is incrementally increased by one, adding the next best clustering to the ensemble. After each addition, the quality of the ensemble is verified. Thereafter the best performing ensemble on the validation set is applied to the independent test set. Note that this procedure is independently repeated for each validation index. In Fig. 1 (b) the procedure of finding the final ensemble members is illustrated, i.e. two validation criteria (BP and C) are plotted as a function of the ensemble size m. In this example the optimal ensemble size is 5 and 15 for C index and BP index, respectively.



(a) RBF kernel validation for all validation indices on one particular embedding.

(b) Ensemble size validation for BP and C index.

Fig. 1. Meta parameter optimization on the independent validation set

4.4 Results and Discussion

In Table 1 the clustering validation indices for both reference systems, i.e. the single clustering algorithms (Sing.) in the original graph domain (GD) and in the embedding vector space (VS), are given for all test data sets. The same criteria are shown for our novel approach, i.e. the ensemble procedure (Ens.) in the embedding vector space. Compared to the first reference system, i.e. the k-medians clustering in the original graph domain, we observe that the ensemble approach results in better clusterings in 15 out of 16 cases. That is, the superiority of our ensemble clustering based on vector space embedded graphs is obvious when compared to the traditional approach in the original graph domain. Regarding the second reference system, the best individual clustering from the ensemble, we observe the following. Our novel approach with a cluster ensemble outperforms this single clustering system in half of the cases (8 out of 16). In five cases both approaches achieve the same criterion value, and in only three cases we observe a deterioration of the clustering quality.

Table 1. Clustering results on the data sets in the graph domain (GD) and the vector space (VS) achieved with single clustering algorithms (Sing.) and the cluster ensemble (Ens.). Bold numbers indicate superior performance over the other systems.

	Dunn			1-C			Rand			BP		
	Sing.		Ens.	Sing.		Ens.	Sing.		Ens.	Sing.		Ens.
Data Set	GD	VS	VS	GD	VS	VS	GD	VS	VS	GD	VS	VS
Letter	0.02	**0.15**	0.12	0.58	**0.94**	**0.94**	0.87	**0.90**	**0.90**	0.23	0.41	**0.45**
COIL	0.13	0.14	**0.15**	0.62	**0.94**	**0.94**	0.69	0.52	**0.80**	0.81	0.62	**0.89**
Fingerprint	**0.21**	0.07	0.09	0.91	0.93	**0.97**	0.32	**0.77**	**0.77**	0.45	**0.67**	**0.67**
Enzymes	0.03	0.03	**0.04**	0.41	**0.96**	0.95	0.49	**0.71**	0.70	0.22	0.27	**0.28**

Evaluating the four validation indices independently of each other, we conclude the following. Regarding the Dunn index, we observe that the clustering based on the ensemble outperforms the single clustering approaches on half of the data sets. Regarding both C and Rand index, the ensemble achieves the best validation criterion value among all procedures on three out of four data sets. Finally, regarding BP our novel approach achieves the best result on all of the four data sets.

Summarizing, with the ensemble procedure proposed in the present paper, the clusterings are in most of the cases more compact and better separable than the clusterings achieved by the single clustering algorithms. Moreover, on all data sets the partitions found by the ensemble are more accurate and consistent according to the ground truth.

5 Conclusions

In the present paper we propose a procedure for building structural cluster ensembles which can be applied to graphs, strings, and trees. The basic idea is to map the structural data in an n-dimensional vector space by means of dissimilarities and prototype selection. In the present study we focus on graph based representation. The fact that our embedding framework crucially depends on the prototype selection strategy, is explicitly utilized for building our ensemble. That is, by means of m randomized prototype sets, a single graph set can be mapped to multiple vector sets. Based on these m vector space embeddings of graphs, a kernel k-means clustering algorithm is applied, resulting in m different clusterings of the same data. Finally, the clusterings are combined by means of a consensus function, interpreting the summed coassociation matrices of each clustering as similarity matrix upon which the final clustering is then defined.

The novel contribution of the proposed approach is threefold. First, it makes the k-means clustering algorithm available to the graph domain. Because of the lack of suitable procedures for computing the mean of a graph population, only k-medians algorithm has been traditionally applied to graphs. Secondly, by means of the embedding procedure we gain the possibility to apply kernel k-means clustering to data that are not spherically structured, as implicitly assumed by the k-means clustering algorithm. Thirdly (and most importantly), by means of a random prototype selection, multiple vector sets can be defined, all representing the same underlying graph set. Through this procedure a multiple clustering system for structural data can be obtained in a straightforward way.

The applicability and performance of our novel approach is tested on four different graph sets with four clustering validation indices. According to the Dunn index our novel approach outperforms the reference systems in two out of four cases. The other indices indicate that our novel approach outperforms the reference systems on most (C, Rand) or even all (BP) data sets.

Potential future work includes an extension to other clustering algorithms and other criteria for evaluating the clustering quality, for example, stability [1].

Acknowledgments

This work has been supported by the Swiss National Science Foundation (Project 200021-113198/1).

References

1. Kuncheva, L., Vetrov, D.: Evaluation of stability of k-means cluster ensembles with respect to random initialization. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(11), 1798–1808 (2006)
2. Jain, A., Murty, M., Flynn, P.: Data clustering: A review. ACM Computing Surveys 31(3), 264–323 (1999)
3. Dudoit, S.: Fridlyand: Bagging to improve the accuracy of a clustering procedure. Bioinformatics 19(9), 1090–1099 (2003)
4. Fred, A., Jain, A.: Combining multiple clusterings using evidence accumulation. IEEE Trans. on Pattern Analysis and Machine Intelligence 27(6), 835–850 (2005)
5. Ayad, H., Kamel, M.: Finding natural clusters using multiclusterer combiner based on shared nearest neighbors. In: Windeatt, T., Roli, F. (eds.) MCS 2003. LNCS, vol. 2709, pp. 166–175. Springer, Heidelberg (2003)
6. Strehl, A., Gosh, J., Cardie, C.: Cluster ensembles–a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3, 583–617 (2002)
7. Englert, R., Glantz, R.: Towards the clustering of graphs. In: Kropatsch, W., Jolion, J. (eds.) Proc. 2nd Int. Workshop on Graph Based Representations in Pattern Recognition, pp. 125–133 (2000)
8. Bunke, H., Dickinson, P., Kraetzl, M., Wallis, W.: A Graph-Theoretic Approach to Enterprise Network Dynamics. In: Progress in Computer Science and Applied Logic (PCS), vol. 24. Birkhäuser, Basel (2007)
9. Mahé, P., Ueda, N., Akutsu, T.: Graph kernels for molecular structures – activity relationship analysis with support vector machines. Journal of Chemical Information and Modeling 45(4), 939–951 (2005)
10. Schenker, A., Bunke, H., Last, M., Kandel, A.: Graph-Theoretic Techniques for Web Content Mining. World Scientific, Singapore (2005)
11. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. Journal of Pattern Recognition and Artificial Intelligence 18(3), 265–298 (2004)

12. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2002)
13. Gärtner, T.: Kernels for Structured Data. World Scientific, Singapore (2008)
14. Pekalska, E., Duin, R.: The Dissimilarity Representation for Pattern Recognition: Foundations and Applications. World Scientific, Singapore (2005)
15. Spillmann, B., Neuhaus, M., Bunke, H., Pekalska, E., Duin, R.: Transforming strings to vector spaces using prototype selection. In: Yeung, D.Y., Kwok, J., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR 2006 and SPR 2006. LNCS, vol. 4109, pp. 287–296. Springer, Heidelberg (2006)
16. Riesen, K., Bunke, H.: Graph classification based on vector space embedding. Int. Journal of Pattern Recognition and Artificial Intelligence (2008) (accepted for publication)
17. Riesen, K., Bunke, H.: Classifier ensembles for vector space embedding of graphs. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 220–230. Springer, Heidelberg (2007)
18. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters 1, 245–253 (1983)
19. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. In: Image and Vision Computing (2008) (accepted for publication)
20. Riesen, K., Bunke, H.: IAM graph database repository for graph based pattern recognition and machine learning. In: da Vitoria, L., et al. (eds.) Structural, Syntactic, and Statistical Pattern Recognition. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)
21. Nene, S., Nayar, S., Murase, H.: Columbia Object Image Library: COIL-100. Technical report, Department of Computer Science, Columbia University, New York (1996)
22. Watson, C., Wilson, C.: NIST Special Database 4, Fingerprint Database. National Institute of Standards and Technology (1992)
23. Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shidyalov, I., Bourne, P.: The protein data bank. Nucleic Acids Research 28, 235–242 (2000)
24. Dunn, J.: Well-separated clusters and optimal fuzzy partitions. Journal of Cybernetics 4, 95–104 (1974)
25. Hubert, L., Schultz, J.: Quadratic assignment as a general data analysis strategy. British Journal of Mathematical and Statistical Psychology 29, 190–241 (1976)
26. Rand, W.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66(336), 846–850 (1971)
27. Riesen, K., Bunke, H.: Kernel k-means clustering applied to vector space embeddings of graphs. In: Prevost, L., Marinai, S., Schwenker, F. (eds.) ANNPR 2008. LNCS (LNAI), vol. 5064, pp. 24–35. Springer, Heidelberg (2008)
28. Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms. John Wiley, Chichester (2004)

Random Ordinality Ensembles: A Novel Ensemble Method for Multi-valued Categorical Data

Amir Ahmad and Gavin Brown

School of Computer Science, University of Manchester,
Manchester, M13 9PL, UK
{ahmada,gbrown}@cs.man.ac.uk

Abstract. Data with multi-valued categorical attributes can cause major problems for decision trees. The high branching factor can lead to data fragmentation, where decisions have little or no statistical support. In this paper, we propose a new ensemble method, *Random Ordinality Ensembles* (ROE), that circumvents this problem, and provides significantly improved accuracies over other popular ensemble methods. We perform a random projection of the *categorical* data into a *continuous* space by imposing *random ordinality* on categorical attribute values. A decision tree that learns on this new continuous space is able to use binary splits, hence avoiding the data fragmentation problem. A majority-vote ensemble is then constructed with several trees, each learnt from a different continuous space. An empirical evaluation on 13 datasets shows this simple method to significantly outperform standard techniques such as Boosting and Random Forests. Theoretical study using an information gain framework is carried out to explain RO performance. Study shows that ROE is quite robust to data fragmentation problem and Random Ordinality (RO) trees are significantly smaller than trees generated using multi-way split.

Keywords: Decision trees, Data fragmentation, Random Ordinality, Binary splits, Multi-way splits.

1 Introduction

Ensembles are a combination of multiple base models for which the final classification depends on the combined outputs of individual models. Classifier ensembles have shown to produce better results than single models, if the classifiers are *accurate* and *diverse* [7,12].

Several different methods based on the principle of data randomization have been proposed to build diverse decision tree ensembles. Some methods manipulate the data, whereas some other methods manipulate the splitting criteria. *Bagging* [3] and *Boosting* [10] introduce randomization by manipulating the distribution of training patterns supplied to each classifier. Random Trees [8] and Random Forests [4] manipulate the splitting criteria to build ensembles of decision trees.

The majority of existing methods [5,13] for decision trees build a tree in a top-down approach and use various *impurity functions* to estimate the quality of the attributes in order to select the best one to *split on*. Whether there should be a *binary split* or *multi-way split* has been a question of extensive research [5,13,9,2]. While multi-way splits produce a more comprehensible tree, they may lead to the *data fragmentation* problem [14], where fine-grained partitioning of the training set at every tree node reduces the number of examples at lower-level nodes. As decisions in the lower levels nodes are based on increasingly smaller fragments of the data, some of them may not have much statistical significance.

Motivated by the advantages of binary decision trees (low data fragmentation) for *multi-valued categorical data* [5,13,9], in the proposed work, we build classifier ensembles of *binary decision trees for datasets consisting of multi-valued categorical attributes.*

The rest of the paper is organized as follows: in the next section, we discuss different binary-split and multi-way split criteria for decision trees. In section 3, we present the *Random Ordinality ensemble* technique. Theoretical study of RO attributes using information gain ratio framework is presented in section 4. The experiments are presented in section 5. The effect of data fragmentation on ROE and sizes of RO trees are studied in section 6. The paper ends with conclusions and future work.

2 Related Work with Split Criteria

In this section, we analyze various split criteria used in decision trees for *multi-valued categorical attributes.*

The CART [5] procedure proposed by Brieman uses the *Gini index* as its splitting criterion. As a multi-way split (for multi-valued categorical attributes) with the Gini index favours those with more values, CART enforces binary splits to overcome this problem. As CART procedure builds binary trees, the values of the categorical attribute at the node have to be divided into two groups. If the number of attribute values is $|A|$ then the number of nontrivial binary splits is given by $2^{(|A|-1)} - 1$. Selecting the best split is computationally expensive. Breiman [5] shows that for two class problems the best split can be found by examining only ($|A|$-1) possibilities.

C4.5 as proposed by Quinlan [13] uses the *information gain ratio* as the splitting criterion. C4.5 builds a binary tree for continuous data. There are two methods in C4.5 to handle multi-valued categorical attributes. In the first, it allows the multi-way split of nodes (one branch for each attribute value). In the second method, it uses a greedy approach to iteratively merge the attribute values into two groups. Another way to obtain a binary split for a multi-valued categorical attribute is to partition the data points using an attribute value [5,9]. In this method, all the data points with that attribute value form one group, whereas the other group is formed with the other examples. Geurts et al. [11] suggest a randomized method to create binary attributes from the multi-valued attributes; they divide the attribute values randomly into the two categories. As in this method the node split decision is taken without considering the output, the classification accuracy of the tree may be poor.

Our method is between the methods proposed by Breiman (searching for the best split) [5] and completely random splits [11]. In RO trees, the best split at each node can be found by examining ($|A|$-1) possibilities. In this next section, we present RO ensembles.

3 Random Ordinality Ensembles

In this section we discuss our proposed method, for producing ensembles of binary decision trees on datasets with *multi-valued categorical attributes.* The handling of categorical attributes is difficult as the category values have no *intrinsic ordering.* For example (*dog, cat, cow*), have no natural order. This is distinct from discrete data, such as (*low, medium, high*), where there is a natural order to the attribute values. We can exploit this property to build an ensemble of binary decision trees. *We solve the node splitting problem under some random constraints.* Our method is between the methods proposed by Breiman (searching for the best split) [5] and completely random splits [11]. To find the best split at each node ($|A|$-1) possibilities are examined. Random constraints used in the proposed method are helpful in building classifier ensembles as the randomization helps in creating diversity. **This technique is based on data manipulation by imposing a random ordinality onto the categorical attribute values.** This implies a random projection of the categorical attributes into a continuous space. Our method is based on data manipulation, so it is not specific to any split criterion—Random Ordinality creates diverse *training datasets.*

3.1 Data Generation Using RO

As there is no natural order given for the categorical attribute values, we can *enforce a random ordinality* on these values. In other words, we create a random projection of categories to a continuous space. We explain our method by using the example data given in column one of Table 1. This data has four attribute values (Cow, Dog, Cat, Rat) for one of its attributes (attribute 1). We assign some integer number (1 to *number of attribute values*) to them randomly such that no two attribute values are assigned the same integer value. For example, we assign Dog = 1, Cow = 2, Rat = 3, Cat = 4 to the attribute values of the first attribute. The enforced ordinality is therefore Dog<Cow<Rat<Cat. We follow the same process for all the multi-valued categorical attributes independently. Our final dataset will be integer-valued, therefore having a natural ordering. Following this method we can generate diverse continuous datasets from the original training dataset.

3.2 Learning

Each decision tree in the ensemble learns on one dataset from the pool of different datasets created by RO. During learning, integer-valued attributes are treated

Table 1. Example of Random Ordinality for a single attribute A_1. The possible values of A_1 have no natural ordering—but can be randomly assigned with an ordinality, as shown by new attributes A_1', and A_1''. In A_1', we have Dog<Cow<Rat<Cat, while in A_1'', we have Rat<Cat<Cow<Dog.

A_1	$\rightarrow$	A_1'	A_1''
Cow	$\rightarrow$	2	3
Dog	$\rightarrow$	1	4
Cow	$\rightarrow$	2	3
Dog	$\rightarrow$	1	4
Rat	$\rightarrow$	3	1
Rat	$\rightarrow$	3	1
Cat	$\rightarrow$	4	2
Cat	$\rightarrow$	4	2

as *continuous attributes.* We have binary splits in the tree as for continuous data attributes the node is split at a threshold value. For our example, we have three possible splits, {(1), (2,3,4)}, {(1,2), (3,4)} and {(1,2,3), (4)}. The best split is decided by the desired split criterion. *We avoid the data fragmentation problem as there is a binary split.* Using this method, it is not necessarily true that we get the best split as shown by Breiman [5]. However, since we want to create an ensemble, different node splits are necessary to create diverse decision trees. Furthermore there is no change in the tree building process so no extra computational cost for the tree building phase. Results of different decision trees in the ensemble are combined using a majority voting scheme to get the final prediction. ROE algorithm is presented in Fig. 1. In the next section, we present theoretical study of RO attributes.

4 Study of RO Attributes in an Information Gain Framework

In RO, new attributes are created by randomly assigning order to different attribute values and treating these new attributes as continuous. The selected splitting criterion is used to decide the best binary split. In this section, we will use the information theoretic framework to discuss whether these attributes are good for classification.

Let D be a 2 class (Y = +1 and Y = -1) dataset with the same number of positive and negative examples. Let A be a multi-valued attribute with cardinality $|A|$ again with uniform prior probability. Half of these values correctly identify the positive class, whereas rest of the values correctly identify the negative class. For example, if attribute values are (a,b,c,d,e,f),

$$p(Y = +1|A = a) = 1, p(Y = +1|A = b) = 1, p(Y = +1|A = c) = 1. \quad (1)$$

$$p(Y = -1|A = d) = 1, p(Y = -1|A = e) = 1, p(Y = -1|A = f) = 1. \quad (2)$$

```
Input- Dataset $T$ with $m$ multi-valued categorical attributes and L size of the ensemble.

Training Phase
for i=1...L do
    Data Generation
    Apply Random Ordinality to generate integer valued dataset $T_i$.
    Learning Phase
    Treat dataset $T_i$ as continuous, and learn decision tree $D_i$.
end for

Testing Phase
For a given data point $\mathbf{x}$
for i=1...L do
    Convert $\mathbf{x}$ to $\mathbf{x}'$ using the ordinality of tree $D_i$.
    Get the prediction for $\mathbf{x}'$ from tree $D_i$.
end for
Combine the results of $L$ decision trees by the chosen combination rule to get the final classification result (we use majority voting method).
```

Fig. 1. Algorithm for Random Ordinality Ensembles (ROE)

We calculate the information gain ratio of different attributes created by RO. We randomly assign order to (a,b,c,d,e,f) and calculate a binary split at each point, the maximum information gain ratio is taken as the information gain ratio associated with this random order. For example, if we assign

$$a < c < f < e < b < d. \tag{3}$$

The maximum information gain ratio is based on the split ((a,c) (f,e,b,d)) and this is taken as the information gain ratio associated with the random order presented in Eq. 3. We calculate the average information gain ratio of different possible random orders of attribute values. We carry out this exercise for attributes with different cardinality, whereas dataset and attribute values have the same properties as discussed above.

We also calculate the information gain ratio of binary splits created by random splitting of attribute values into two groups. Results are presented in table 2. Results indicate that the average gain ratio of attribute created using RO and the gain ratio of multi-valued attributes are quite similar, whereas random splits do not create good splits. As the cardinality of the attribute increases, the average information gain ratio of RO attributes decreases. The same is true for the multi-way split as the value of normalizing factor ($\log_2 |A|$) increases. This suggests that on average we are creating binary splits from multi-valued categorical attributes that have similar information gain ratio. The theoretical study suggests that for multi-valued categorical attributes with certain properties, **the information gain**

Table 2. Gain ratio of attributes with different numbers of attribute values

Cardinality $\|A\|$ of the attribute A	Number of random attributes created	Average gain ratio for RO attributes (s.d.)	Average gain ratio for attributes with random split (s.d.)	Gain ratio for multi-way split
4	10^4	**0.59(0.29)**	0.37(0.26)	0.50
6	10^4	**0.47(0.20)**	0.20(0.24)	0.39
8	10^6	**0.40(0.16)**	0.13(0.18)	0.33
10	10^7	**0.35(0.12)**	0.10(0.14)	0.30
12	10^7	**0.32(0.10)**	0.08(0.11)	0.28
14	10^7	**0.29(0.09)**	0.06(0.09)	0.26

ratio of a binary split with some random constraints may be equal to or greater than a multi-way split.

In the next section, we present the comparative study of ROE against the other ensemble methods.

5 Empirical Evaluation

A study was carried out to compare the performance of ROE with Bagging [3], AdaBoostM1 [10] and Random Forest [4]. We created two types of RO ensembles. In the first, ROE with J48, we used the J48 (the WEKA [15] implementation of C4.5 as the base classifier (with the unpruned option)), which uses multi-way splits for multi-valued categorical attributes as per default. In the second, ROE with RS, we used *Random Trees*[15] as the base classifier. *Random Trees* [15] constructs a tree that considers K random features at each node. In other words, we combine the benefits of attribute randomization of *Random Subspaces* (RS) with *Random Ordinality.* We carried out experiments with Bagging and AdaBoost.M1 [10] using J48 (unpruned) as the base model, and Random Forests (WEKA implementations of these ensemble method were used). The sizes of the ensembles were set at 50 for these experiments. K (number of attributes to randomly investigate) is taken as the half of the attributes for Random Tree. Default settings were used for the rest of the parameters. The experiments were conducted following the 5×2 cross-validation [6]. The original test proposed by Dietterich [6] to compare the performance of classifiers suffers from low replicability. Alpaydin [1] propose a modification to the 5×2 cross-validation F test. We used this test for our experiments. We considered a confidence level of 95% for this test. Table 3 presents classification errors of different ensemble methods on different datasets.

Results suggest that, with the exception of Monks1 data, the performance of ROE with J48 is either statistically similar or better than that of other popular ensemble methods. *The performance of ROE with RS is either statistically similar or better than that of other popular ensemble methods for all datasets.* For Monks1 data the performance of ROE with J48 was poor. We discuss this dataset in detail to understand the limitations of RO ensembles.

Table 3. Classification error in % for different ensembles; bold numbers indicate best performance. Comparative results are presented *ROE with J48/ROE with RS* in bracket (if performance of these ensembles are different). '+/-' shows that performance of ROE is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between ROE and that algorithm.

Dataset	RO with J48 ensemble	RO with RS ensemble	Bagging	AdaBoostM1	Random Forest	Single Tree (J48)
Promoter	13.1	**12.8**	15.5	19.6	13.4	28.5(+/+)
Hayes-Roth	16.9	**15.9**	22.8(+/+)	23.1(+/+)	22.2(+/+)	25.3(+/+)
Breast Cancer	30.3	30.1	**29.9**	35.6	32.4	35.9
Monks1	18.3	**1.5**	5.8(-/Δ)	5.9(-/Δ)	3.3(-/Δ)	15.9(-/+)
Monks2	33.9	**30.9**	46.9(+/+)	47.5(+/+)	50.4(+/+)	49.6(+/+)
Monks3	0	0	0	0	0	0
Balance	**19.6**	20.0	29.6(+/+)	30.3(+/+)	26.9(+/+)	31.4(+/+)
Soyalarge	8.8	**7.3**	8.2	**7.3**	7.9	9.7
Tic-tac-toe	6.6	**3.4**	10.0(+/+)	3.5	8.6(Δ/+)	18.4(+/+)
Car	**4.1**	4.2	8.3(+/+)	5.9(+/+)	8.3(+/+)	9.2(+/+)
DNA	4.5	4.4	6.2(+/+)	5.1	5.8	8.9(+/+)
Mushroom	0.1	0.1	**0**	**0**	**0**	**0**
Nursery	1.0	**0.9**	2.8(+/+)	1.3(+/+)	2.6(+/+)	3.6(+/+)
RO with J48 win/draw/lose			7/5/1	5/7/1	5/7/1	9/3/1
RO with RS win/draw/lose			7/6/0	5/8/0	6/7/0	9/4/0

Monks1 dataset has six attributes and two classes. The classification is Y = 1, if $(x_1 = x_2) \vee (x_5 = 1)$. All the other data points belong to class 2. When we treat data as continuous, the first concept $(x_1 = x_2)$ is a diagonal concept. J48 trees are restricted to *orthogonal* decision boundaries. In other words, decision trees divide the input attribute space into rectangular regions whose sides are perpendicular to the attribute axis. Decision trees have a representational problem because of this orthogonal property; they cannot learn diagonal concepts properly. Ensembles of decision trees solve this problem, as combined results of decision trees produce a good approximation of a diagonal concept [7]. The quality of the approximation depends on the diversity of decision trees in the ensemble. RO with RS trees are more diverse as compared to RO with J48 trees. Hence, ROE with RS can learn this diagonal concept in Monk1 data better than ROE with J48.

Building a good ensemble depends on the creation of diverse decision trees. We create diverse decision trees by imposing random ordinality to categorical attributes values that in turn create different node splits. The diversity in node splits is the key for diverse decision trees. If we have $|A|$ attribute values, these attributes will be present in different trees in different order, the possible number of different splits from these attribute values is $2^{(|A|-1)} - 1$. If $|A|$ is small, there is a large possibility that different trees have same node splits, and we may not get very diverse trees. Tic-Tac-Toe data has only 3 attribute values for each attribute.

Hence, there is only three possible node splits for each attribute (we are taking the case when all the attribute values are present in the node). Different trees can have one of the three possible node splits for a attribute. It means that there is a large possibility that different trees have same node splits. In this condition, the trees in the ensemble will not be very diverse. When we combine the attribute randomization of RS with RO, we observe a large improvement in the classification error as compared to ROE with J48 (the average error reduced from 6.6% to 3.4%). Better diversity of ROE with RS is the reason for this improvement. In the next section, we present the various studies to analyze RO trees and RO ensembles.

6 Study of RO Ensembles and RO Trees

One of the motivation of ROE is that it avoids data fragmentation problem. In this section we study the effect of data fragmentation on ROE and RO tree sizes.

6.1 Study of Data Fragmentation for ROE

Data fragmentation may affect the performance of decision trees. We have carried out a controlled experiment to see how different ensemble methods perform with respect to the number of attribute values. For this purpose, we selected two pure continuous datasets; Segment and Vehicle. We converted these datasets into categorical datasets using equal width discretization. We studied various ensemble methods on these discretized datasets; varying the numbers of bins to see its effect on different ensemble methods. We performed five replications of a two-fold cross-validation. The results (Fig. 2) suggest that classification errors of RO ensembles are relatively unaffected. When we increase the number of bins we have a small number of points in every bin; that leads to badly estimated probabilities and poor generalization. Whereas, RO ensembles have binary decision trees so they are more robust to the data fragmentation problem.

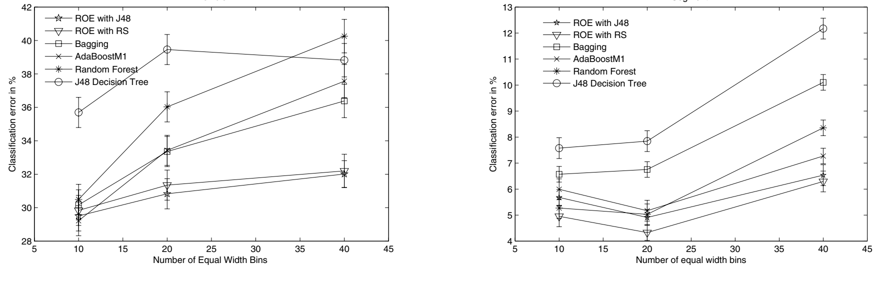


Fig. 2. Effect of equal width discretization on various ensemble methods for Vehicle and Segment datasets. RO resists fragmentation as space grows.

Table 4. The average sizes of RO trees and multi-split J48 trees for different datasets

Name of dataset	Size of the training data	The average number of leaves/size of RO trees (J48)	The average number of leaves/size of multi-way split J48 trees
Car	864	54/107	127/174
DNA	1587	76/151	211/281
Tic-Tac-Toe	479	49/97	92/142
Promoter	53	6/11	13/17

6.2 RO Tree Sizes

Smaller trees have greater statistical evidence at the leaves. Motivated by Occams Razor, small trees are preferable. As RO trees have binary splits RO trees are more likely to have smaller sizes than that of multi-split decision trees. We studied RO tree sizes for various datasets; Car, DNA, Tic-Tac-Toe and Promoter. The experiments were conducted following the 5×2 cross-validation and 50 RO trees are created in each run.

In the table 4, we present the average sizes of RO trees (J48 decision trees created using datasets generated by RO method) and normal multi-split J48 decision trees for different datasets. For all the datasets, RO trees are smaller that normal multi-split J48 decision trees. For example, for DNA dataset, the average size of RO trees is 151 whereas the average size of normal multi-split J48 decision trees is 281. **These results indicate that RO helps in creating smaller decision trees**.

7 Conclusion

In this paper, we have presented a new ensemble method to build *diverse binary decision trees for datasets consisting of multi-valued categorical attributes*. We convert categorical attributes into continuous attributes by randomly assigning integer values to categorical attribute values. As the transformation to continuous data is random, diverse datasets are created. When a decision tree is constructed by treating these new attributes as continuous ones, we have binary splits at the nodes giving binary decision trees. The theoretical study suggests that for multi-valued categorical attributes with certain properties, the information gain ratio of a binary split of RO attributes may be equal to or greater than a multi-way split. We create two types of ensembles using RO. In the first, we use J48 (the WEKA [15] implementation of C4.5) as the base model for the ensemble. In the second, we combine the attribute randomization of Random Subspaces with Random Ordinality. The comparative study on 13 different datasets from the UCI repository suggest that ROE significantly outperform other popular ensemble methods in terms of test error. The study shows that ROE avoids the data fragmentation problem and RO trees are significately smaller than multi-way split trees. ROE is easy to implement and parallel implementation of ROE is also possible.

In this present work, we imposed random ordinality to each attribute independently. In future we will also take interdependencies of attributes into consideration while imposing random ordinality. The "take-home" message of this paper is that, when categorical attribute values have no *intrinsic order*, this property can be exploited to build a successfully performing ensemble of diverse binary decision trees.

References

1. Alpaydin, E.: Combined 5 x 2 cv f Test Comparing Supervised Classification Learning Algorithms. Neural Computation 11(8), 1885–1892 (1999)
2. Bratko, I., Kononenko, I.: Learning Diagnostic Rules from Incomplete and Noisy Data, Seminar on AI Methods in Statistics, London (1986)
3. Breiman, L.: Bagging Predictors. Machine Learning 24(2), 123–140 (1996)
4. Breiman, L.: Random Forests. Machine Learning 45(1), 5–32 (2001)
5. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth International Group, CA (1985)
6. Dietterich, T.G.: Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation 10, 1895–1923 (1998)
7. Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
8. Dietterich, T.G.: An Experimental Comparison of Three Methods for Constructing Ensembles of Decision trees: Bagging, Boosting, and randomization. Machine Learning 40(2), 1–22 (2000)
9. Fayyad, U.M., Irani, K.B.: The Attribute Selection Problem in Decision Tree Generation. In: Proc. AAAI 1992. MIT Press, Cambridge (1992)
10. Freund, Y., Schapire, R.E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
11. Geurts, P., Ernst, D., Wehenkel, L.: Extremely Randomized Trees. Machine Learning 63(1), 3–42 (2006)
12. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
13. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
14. Vilalta, R., Blix, G., Rendell, L.: Global Data Analysis and the Fragmentation Problem in Decision Tree Induction. In: Proceedings of the 9th European Conference on Machine Learning, pp. 312–328 (1997)
15. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)

True Path Rule Hierarchical Ensembles

Giorgio Valentini

DSI, Dipartimento di Scienze dell' Informazione,
Università degli Studi di Milano,
Via Comelico 39, 20135 Milano, Italia
valentini@dsi.unimi.it

Abstract. Hierarchical classification problems gained increasing attention within the machine learning community, and several methods for hierarchically structured taxonomies have been recently proposed, with applications ranging from classification of web documents to bioinformatics. In this paper we propose a novel ensemble algorithm for multilabel, multi-path, tree-structured hierarchical classification problems based on the true path rule borrowed from the Gene Ontology. Local base classifiers, each specialized to recognize a single class of the hierarchy, exchange information between them to achieve a global "consensus" ensemble decision. A two-way asymmetric flow of information crosses the tree-structured ensemble: positive predictions for a node influence its ancestors, while negative predictions influence its offsprings. The resulting *True Path Rule* hierarchical ensemble is applied to the prediction of gene function in the yeast, using the FunCat taxonomy and biomolecular data obtained from high-throughput biotechnologies.

1 Introduction

Several interesting real-world classification problems are characterized by hierarchical relationships between classes [1, 2, 3]. These problems come from different fields, ranging from textual classification of web content [1, 2], to gene function prediction in bioinformatics [3, 4], and share the common property that a certain general class may be further specified by more refined classes at different levels of an overall hierarchy. For instance, in the *FunCat* taxonomy [5] the general class "metabolism" has several child classes, such as "amino acid metabolism", "C-compound and carbohydrate metabolism", "lipid and fatty acid metabolism" and others that provide more detailed specifications and subdivisions of the parent class. Moreover each child class, e.g. "amino acid metabolism", can be further subdivided in "metabolism of the aspartate family", "metabolism of the cysteine - aromatic group" and so on, thus resulting in a complex hierarchy divided at multiple levels.

Several hierarchical algorithms have been proposed in the literature, with different characteristics and purposes, considering for instance methods restricted to multilabels with single and no partial paths [1, 6], or other methods extended to multiple and also partial paths [2, 7]. Nevertheless, algorithms that explicitly take into account the relationships between the classes of the structured hierarchy received much less attention.

In particular in this paper we propose a hierarchical ensemble algorithm, by which classifications of positive examples in child nodes influence the prediction of the parent node in a recursive way, while negative predictions in a node influence the prediction in the descendant nodes. This general behaviour is a consequence of the *true path rule*, a term borrowed from the Gene Ontology [8]: according to this rule, if an example belongs to a class, it belongs to all its ancestors, and if does not belong to a class it does not belong to all its offsprings.

In the next section the main motivations and characteristics of the proposed ensemble algorithm are presented and discussed. Then in Sect. 3 we test the proposed method on a complex hierarchical gene function prediction problem, using the FunCat taxonomy and bio-molecular data obtained from public databases, and discuss some drawbacks and possible enhancements of the proposed hierarchical ensemble approach. The conclusions end the paper.

2 An Ensemble Algorithm Based on the True Path Rule

2.1 Definitions and Notation

We consider a multiclass multilabel classification problem where the classes are structured according to a given hierarchy.

More precisely, an example x can be assigned to 1 or more classes of the set $\Omega = \{\omega_1, \omega_2, \ldots, \omega_m\}$. The assignments are coded through a vector of multilabels $\mathbf{y} =< y_1, y_2, \ldots, y_m >\in \{0,1\}^m$, by which if x belongs to class ω_j, then $y_j = 1$, otherwise $y_j = 0$.

The classes are structured according to a hierarchy and can be represented by a directed graph, where nodes correspond to classes, and arcs to relationships between classes. Considering that each node corresponds to a class, the node corresponding to the class ω_i may be simply denoted by i. We denote by child(i) the set of children nodes of i, while par(i) represents the set of the parents of node i. Moreover $y_{child(i)}$ denotes the labels of the children classes of node i and analogously $y_{par(i)}$ denotes the labels of the parent classes of i.

A classifier $D : X \rightarrow \{0,1\}^m$ computes the multilabel associated to each example $x \in X$, and $d_i(x) \in \{0,1\}$ is the label predicted by the classifier for class ω_i. For the sake of simplicity if there is no ambiguity we represent $d_i(x)$ simply by d_i.

2.2 The True Path Rule

The proposed algorithm is inspired by the ”true path rule” that characterizes the hierarchy of the gene functional classes of both the *Gene Ontology (GO)* [8] and *FunCat* [5] taxonomies:

> “If the child term describes the gene product, then all its parent terms must also apply to that gene product”

This means that if a gene is annotated with a specific functional term (functional class), then it is annotated with all the ”parent” classes, and with all its ancestors

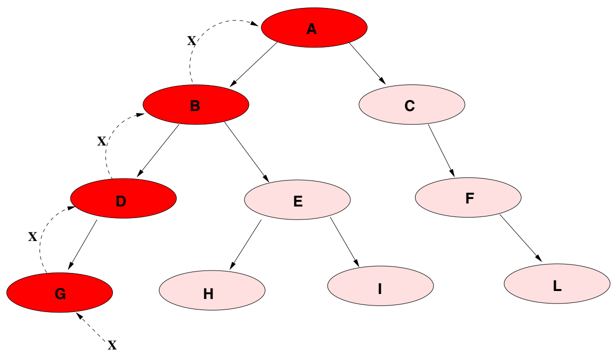


Fig. 1. True path rule: if example x belongs to class G then it belongs also to class D, B and A. On the contrary if an example x does not belong to class C it cannot belong to class F or L.

in a recursive way. On the contrary if a gene is not annotated to a a class, it cannot be annotated to its offsprings. (Fig. 1).

From the "true path rule", for a given example x, considering the parents of a given node i, the following rules can be immediately deduced:

$$\begin{cases} y_i = 1 \Rightarrow y_{par(i)} = 1 \\ y_i = 0 \nRightarrow y_{par(i)} = 0 \end{cases} \tag{1}$$

As a consequence a classifier that respects the true path rule needs to obey the following rules:

$$\begin{cases} d_i = 1 \Rightarrow d_{par(i)} = 1 \\ d_i = 0 \nRightarrow d_{par(i)} = 0 \end{cases} \tag{2}$$

On the other hand, considering the children of a given node i, the following rules can be immediately deduced:

$$\begin{cases} y_i = 1 \nRightarrow y_{child(i)} = 1 \\ y_i = 0 \Rightarrow y_{child(i)} = 0 \end{cases} \tag{3}$$

and a classifier that respects the true path rule needs to obey the following rules:

$$\begin{cases} d_i = 1 \nRightarrow d_{child(i)} = 1 \\ d_i = 0 \Rightarrow d_{child(i)} = 0 \end{cases} \tag{4}$$

From eq. 1 and 3 we can observe an asymmetry in the rules that govern the assignments of positive and negative labels. Indeed we have a propagation of positive labels from bottom to top of the hierarchy (eq. 1), and a propagation of negative labels from top to bottom (eq. 3). On the contrary negative labels cannot propagate from bottom to top, and positive predictions cannot propagate from top to bottom.

2.3 The Main Ideas behind the Algorithm

We can design a hierarchical classifier that uses the predictions made at each node by local "base" classifiers and puts together their decisions to realize an ensemble that obeys the "true path rule". More precisely the basic ideas behind the *true path rule ensemble algorithm* are the following:

1. A set of base classifiers associated to each class/node of the graph provides a local decision about the assignment of a given example to a given node.
2. Positive decisions for a node influence the decisions made by the parent nodes in a recursive way (that is a positive decision influences the parent and may propagate from bottom to top across the graph). On the contrary negative decisions do no affect decisions of the parent node (that is they do not propagate from bottom to up, eq. 2).
3. If the classifier takes a negative prediction for a given node (taking into account the local decision of its descendants), it in turns set to negative all its descendants, to preserve the consistency of the hierarchy according to the true path rule. On the contrary positive decisions do not influence decisions of child nodes (eq. 4).

The decision of the ensemble classifier is thus the result of the local predictions made by the base classifiers associated to each node modified in order to take into account positive predictions that comes from the bottom of the graph and negative predictions that comes from the top of the graph.

We propose an algorithm for tree-structured graphs that scans the tree from bottom to top through a per level traversal of the tree. Base classifiers estimate local probabilities $\hat{p}_i(x)$ that a given example x belongs to class ω_i, and the ensemble corrects the local probabilities to estimate the "consensus" probability $p_i(x)$. More precisely, given the local estimates of the probabilities $\hat{p}_j(x)$ made by the base classifiers across the tree T of the m classes, the probability that an example x belongs to class ω_i is:

$$p_i(x) = P(\omega_i | x, T, \hat{p}_j(x), 1 \leq j \leq m) \tag{5}$$

2.4 The Hierarchical Ensemble Algorithm

The algorithms starts to train the m base learners (one for each node/class of the hierarchy); each trained classifiers computes an estimate of the local probabilities $\hat{p}_j(x)$. The core of the algorithm is represented by the evaluation phase, where the ensemble provides an estimate of the "consensus" global probability $p_i(x)$. A detailed representation of the evaluation phase of the algorithm is given in Algorithm 1. In the algorithm there are two main for loops: the external for (from row 1 to 26) handles a per level bottom-up traversal of the tree, while the internal (from row 2 to 25) scans the nodes at each level. If a node is a leaf (row 3), then the consensus probability p_i is equal to the local probability $\hat{p}_i(x)$. Note that a positive decision is taken if $p_i(x)$ is larger than a threshold t (row 5): a natural choice for t is 0.5. If a node is not a leaf (row 10), at first the

Algorithm 1. True Path Rule (TPR) hierarchical ensemble

Input:
- a test example x
- tree T of the m hierarchical classes
- set of m classifiers (one for each node) each predicting $\hat{p}_i(x)$, $1 \leq i \leq m$

```
1: for all levels $k$ of $T$ from bottom to top do
2:    for all nodes $i$ at level $k$ do
3:       if $i$ is a leaf then
4:          $p_i(x) \leftarrow \hat{p}_i(x)$
5:          if $p_i(x) > t$ then
6:             $d_i(x) \leftarrow 1$
7:          else
8:             $d_i(x) \leftarrow 0$
9:          end if
10:      else
11:         $\phi(x) \leftarrow \{j | j \in \text{child}(i), d_j(x) = 1\}$
12:         $p_i(x) \leftarrow \frac{1}{1+|\phi(x)|} \left( \hat{p}_i(x) + \sum_{j \in \phi(x)} p_j(x) \right)$
13:         if $p_i(x) > t$ then
14:            $d_i(x) \leftarrow 1$
15:         else
16:            $d_i(x) \leftarrow 0$
17:            for all $j \in \text{subtree}(i)$ do
18:               $d_j(x) \leftarrow 0$
19:               if $p_j(x) > t$ then
20:                  $p_j(x) \leftarrow t$
21:               end if
22:            end for
23:         end if
24:      end if
25:   end for
26: end for
```

Output:
- the ensemble decisions $d_i(x) = \begin{cases} 1 & \text{if } x \text{ belongs to node } i \\ 0 & \text{otherwise} \end{cases}$
- the probabilities $p_i(x)$ that x belongs to the node $i \in T$

set $\phi(x)$ collects all the children nodes for which we have a positive prediction, and the consensus probability p_i of the ensemble is computed by considering both the local estimate of the probability $\hat{p}_i$ and the probabilities computed by the children nodes for which a positive decision has been taken (row 12). Note that in case of a negative decision for the node i, all the classes belonging to the subtree rooted at i are set to negative (rows 17-18). The algorithm provides both the multilabels associated to the example x and the probabilities p_i that a given example belongs to the class i, $1 \leq i \leq m$.

3 Experimental Results

3.1 Hierarchical Classification of Functional Classes of Genes

We considered the functional classification of yeast genes for a large number of classes structured according to the *FunCat* (Functional Catalogue), a hierarchically tree-structured, controlled classification system enabling the functional description of proteins from any organism [5].

We selected only the genes annotated to FunCat (funcat-2.1 scheme), available from the MIPS web site (`http://mips.gsf.de/projects/funcat`), using the *Hcgene* R package [9]. We also removed the genes annotated only with the "99" FunCat class ("UNCLASSIFIED PROTEINS") and selected classes with at least 20 positive examples, in order to get a not too small set of positive examples for training. The resulting tree has a depth equal to 5 and includes about 200 functional classes. Different strategies can be chosen to select negative examples for each functional class [10, 9]. In this work negative examples for each class have been selected in such a way that they are not annotated for the class, but belong to the parent class (i.e. positive for the parent class). In this way only negative examples that are not too dissimilar to the positive ones are selected.

3.2 Data Sets

We chose four different types of bio-molecular data obtained from high-throughput bio-technologies and available from public databases or from literature. The main characteristics of the data we used in our experiments are summarized in Tab. 1.

Proteins are constituted by structured and functionally characterized regions usually referred as domains joined by unstructured regions named loops. To capture this source of functional information we considered the E-value assigned to each gene product by a collection of profile-HMMs, each of which trained on a specific domain family, using data from the *Pfam* (Protein families) database [11]. The E-values have been obtained by means of the HMMER software toolkit [12].

Phylogenetic data have been obtained through BLAST searches [13]: each feature corresponds to the negative logarithm of the lowest E-value reported by BLAST version 2.0 in a search against a complete genome, with negative values (corresponding to E-values greater than 1) truncated to 0 [14].

We merged the gene expression experiments of Spellman et al. (gene expression measures relative to 77 conditions) [15] with the transcriptional responses of

Table 1. Data sets

Data set	n. examples	n. feat.	n.classes
Protein domain	3529	5724	211
Phylogenesis	2445	24	187
Gene expression	4532	250	230
PPI - BioGRID	4531	5367	232

yeast to environmental stress (173 conditions) by Gasch et al. [16], thus obtaining real-valued vector data with 250 features.

Finally we downloaded protein-protein interaction (PPI) data from the *BioGRID* database, that collects PPI data from both high-throughput studies and conventional focused studies [17]. Data are binary: they represent the presence or absence of protein-protein interactions.

3.3 Experimental Setup

For each data set we evaluated the performance of three different ensembles: the *Flat* ensemble, that does not take into account the hierarchical structure of the data, the *Hierarchical Top-Down* and the proposed *True Path Rule (TPR)* Hierarchical Bottom-Up ensemble. The classical hierarchical Top-down algorithm classifies an example x, where $d_i(x)$ is the classifier decision at node i and $root(T)$ denotes the set of nodes at the first level of the tree T, in the following way:

$$y_i = \begin{cases} d_i(x) & \text{if } i \in root(T) \\ d_i(x) & \text{if } i \notin root(T) \wedge y_{par(i)} = 1 \\ 0 & \text{if } i \notin root(T) \wedge y_{par(i)} = 0 \end{cases}$$

As base learners we used 2^{nd} and 3^{rd} degree polynomial SVMs. The probabilistic output of the SVMs composing TPR ensembles has been computed using the sigmoid fitting proposed in [18].

Considering the large unbalance between positive and negative examples available for each class, we evaluated the performance of the ensembles through the F-measure, i.e. the harmonic mean between precision and recall, by applying for each data set 5-fold cross-validation techniques. We performed a limited model selection for the base learners, by applying a grid search only to the first level nodes (classifiers) of the tree (the nodes closest to the root), and then we extended the resulting best model parameters to all the other classifiers of the tree.

3.4 Results

Results of the comparison between Flat, Top-down and True Path Rule hierarchical ensembles are summarized in Tab. 2. The table reports the average F-measure across classes, using the same 0/1 loss for each class of the hierarchy. Data in bold denote results for an ensemble better than both the other two (at 0.05 significance level), according to the 5-fold cross-validated paired t-test [19]. True Path Rule ensembles achieve significantly better results with respect to both Flat and Hierarchical top-down ensembles: only with Gene expression data Top-down ensembles perform better, even if the difference is not statistically significant.

Looking at Tab. 3 we can observe that the better results of TPR ensembles are due to a better balancing between precision and recall. Indeed on the average the higher recall is obtained by the Flat ensemble, while the higher average precision by the Top-down ensembles (Tab. 3). In both cases the recall and precision of the

Table 2. Average F-measure across FunCat classes: comparison between Flat, Top-down and TPR (true path rule) ensembles

Data set	**Flat**	**Top-down**	**TPR**
Protein domain	0.0976	0.1246	**0.1590**
Phylogenetic	0.0204	0.0005	**0.0708**
Gene expression	0.0882	0.1139	0.1058
PPI - BioGRID	0.0396	0.0255	**0.1257**
Average across data	0.0614	0.0661	**0.1153**

Table 3. Average Precision and Recall across FunCat classes: comparison between Flat, Top-down and TPR (true path rule) ensembles

	Flat		**Top-down**		**TPR**	
Data set	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Protein domain	0.1133	0.3256	0.3370	0.0800	0.1488	0.2395
Phylogenetic	0.1288	0.2095	0.0103	0.0002	0.1050	0.0853
Gene expression	0.0669	0.3772	0.1518	0.0961	0.0757	0.2777
PPI - BioGRID	0.1462	0.2282	0.2235	0.0145	0.1862	0.1204
Average across data	0.1138	0.2851	0.1806	0.0477	0.1289	0.1807

TPR ensemble is on the middle, but results in a larger F-measure. Nevertheless, for real applications to gene function prediction, the precision is actually too low to be useful in practice. Indeed in real applications an "in silico" prediction needs to be validated by "in vitro" biological functional validation, and we need a reasonably high precision to justify the more expensive biological validation.

Note that here we consider the average precision, recall, and F-measure across classes, and hence we may obtain an average F-measure that is lower of both the average precision and recall.

Even if the average accuracy across classes is quite high (for both Hierarchical Top-down and TPR ensembles is larger than 90%, while for Flat is about 75%, data not shown), note that this results is not so significant, considering the large unbalance between positive and negative examples for most functional classes. On the contrary the F-measure is quite low: the average across data sets is only 0.1153 for TPR ensembles and this result is halved with both Flat and Top-down ensembles (Tab. 2).

These relatively poor results are due to the intrinsic complexity of the hierarchical multiclass multilabel classification of genes [20]. In many cases bio-molecular data obtained through complex bio-technologies are affected by a relatively high degree of noise. Moreover, usually each data set can provide useful information only for a subset of classes, while for others may be substantially uninformative. It is well-known that by combining multiple sources of data we can substantially improve the results [14, 3], and we may expect substantial improvements by applying data fusion techniques with TPR ensembles.

Another important problem is the local model selection of each base learner. In the experiments, for computational complexity reasons, we applied a relatively moderate model selection strategy limited only to the first level of the tree hierarchy (16 nodes/classes out of more than 200). By applying a computational intensive model selection through internal cross validation we may expect a further improvement of the results of TPR ensembles.

4 Conclusions

In this work we presented a novel ensemble algorithm for multiclass multilabel hierarchical classification problems. The training phase is straightforward (even if computationally intensive), but the core of the algorithm is represented by the evaluation phase. At this stage the base classifiers associated to each node of the tree exchange information in an asymmetric way from bottom to top and top to bottom: positive predictions affect the decisions at "higher level" nodes (i.e. ancestor nodes), while negative predictions affect offsprings, according to the *true path rule* borrowed from the Gene Ontology.

Even if this algorithm has been conceived for the prediction of the function of genes at genome-wide level, it is sufficiently general to be applied in other similar hierarchical problems in different fields and contexts.

The preliminary experimental results show that TPR ensembles are competitive with respect to both classical Flat and Hierarchical Top-down ensembles, and suggest also further directions to improve the basic TPR algorithm. For instance, considering that the decision for a class is influenced only by positive decisions of its offsprings, an ongoing research line consists in explicitly balancing the weigth of the local predictor with respect to that of its children: in this way we could tune the precision and the recall of the ensemble. Moreover, by introducing model selection strategies at each node and data fusion techniques to exploit multiple sources of biomolecular data, we may expect to substantially improve the overall performance of the ensemble.

Acknowledgments

The author would like to thank the anonymous reviewers for their comments, and gratefully acknowledges partial support by the PASCAL2 Network of Excellence under EC grant no. 216886. This publication only reflects the author's views.

References

[1] Dumais, S., Chen, H.: Hierarchical classification of web content. In: Proc. of the 23rd ACM Int. Conf. on Research and Development in Information Retrieval, pp. 256–263. ACM Press, New York (2000)

[2] Rousu, J., et al.: Learning hierarchical multi-category text classification models. In: Proc. of the 22nd ICML, pp. 745–752. OmniPress (2005)

[3] Barutcuoglu, Z., Schapire, R., Troyanskaya, O.: Hierarchical multi-label prediction of gene function. Bioinformatics 22, 830–836 (2006)
[4] Guan, Y., et al.: Predicting gene function in a hierarchical context with an ensemble of classifiers. Genome Biology 9 (2008)
[5] Ruepp, A., et al.: The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucl. Ac. Res. 32, 5539–5545 (2004)
[6] Dekel, O., Keshet, J., Singer, Y.: Large margin hierarchical classification. In: Proc. of the 21st ICML, pp. 209–216. Omnipress (2004)
[7] Cesa-Bianchi, N., Gentile, C., Zaniboni, L.: Hierarchical classification: Combining Bayes with SVM. In: Proc. of the 23rd ICML, pp. 177–184. ACM Press, New York (2006)
[8] The Gene Ontology Consortium: Gene ontology: tool for the unification of biology. Nature Genet. 25, 25–29 (2000)
[9] Valentini, G., Cesa-Bianchi, N.: Hcgene: a software tool to support the hierarchical classification of genes. Bioinformatics 24, 729–731 (2008)
[10] Ben-Hur, A., Noble, W.: Choosing negative examples for the prediction of protein-protein interactions. BMC Bioinformatics 7 (2006)
[11] Finn, R., et al.: The Pfam protein families database. Nucl. Ac. Res. 36, D281–D288 (2008)
[12] Eddy, S.: Profile hidden markov models. Bioinformatics 14, 755–763 (1998)
[13] Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D.: Basic local alignment search tool. Journal of Molecular Biology 215 (1990)
[14] Pavlidis, P., Weston, J., Cai, J., Noble, W.: Learning gene functional classification from multiple data. J. Comput. Biol. 9, 401–411 (2002)
[15] Spellman, P., et al.: Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomices cerevisiae by microarray hybridization. Mol. Biol. Cell 9, 3273–3297 (1998)
[16] Gasch, P., et al.: Genomic expression programs in the response of yeast cells to environmental changes. Mol. Biol. Cell 11, 4241–4257 (2000)
[17] Stark, C., et al.: BioGRID: a general repository for interaction datasets. Nucl. Ac. Res. 34, D535–D539 (2006)
[18] Lin, H., Lin, C., Weng, R.: A note on Platt's probabilistic outputs for support vector machines. Machine Learning 68, 267–276 (2007)
[19] Dietterich, T.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation 10, 1895–1924 (1998)
[20] Pena-Castillo, L., et al.: A critical assessment of Mus musculus gene function prediction using integrated genomic evidence. Genome Biology 9 (2008)

A Study of Semi-supervised Generative Ensembles

Manuela Zanda and Gavin Brown

School of Computer Science, University of Manchester, UK
{zandam,gbrown}@cs.man.ac.uk

Abstract. Machine Learning can be divided into two schools of thought: generative model learning and discriminative model learning. While the MCS community has been focused mainly on the latter, our paper is concerned with questions that arise from ensembles of generative models. Generative models provide us with neat ways of thinking about two interesting learning issues: model selection and semi-supervised learning. Preliminary results show that for semi-supervised low-variance generative models, traditional MCS techniques like Bagging and Random Subspace Method (RSM) do not outperform the single classifier approach. However, RSM introduces diversity between base classifiers. This starting point suggests that diversity between base components has to lie within the *structure* of the base classifier, and not in the dataset, and it highlights the need for novel generative ensemble learning techniques.

1 Introduction

In the past few years, the MCS community has mainly focused on *supervised* problems, that is, learning scenarios where classifiers are trained on *labelled* examples. Nevertheless, many real applications are nowadays characterised by two *contrasting factors*, namely *the need for large quantities of labelled data* to design supervised classifiers with high accuracy, and *the difficulty and cost of collecting such data*.

A possible answer to this accuracy/labelling dilemma is to consider *semi-supervised* algorithms, that is, techniques which are able to learn from a small amount of labelled data together with a large amount of unlabelled data [1]. The majority of the work done so far has been concerned with ensembles of semi-supervised *discriminative* models, where some external procedure is responsible for labelling the unlabelled data before base classifiers can learn from them [2,3,4,5,6].

Generative models are algorithms that can learn from labelled and unlabelled data [1]. There are very few examples of semi-supervised generative ensembles [7,8], and so far there is still no common understanding of the way unlabelled data affects ensembles of generative models.

This paper is an attempt to further investigate semi-supervised ensembles of generative models. Generative and discriminative approaches are two ways of solving the same problem (Sec. 2). A comparison of ensemble techniques shows that the way they make use of unlabelled data is different (Sec. 3); moreover they provide different levels of understanding in terms of model mismatch (Sec. 4). As generative models have not been explored in the MCS community, we present a preliminary experimental analysis (Sec. 5). Our results show that for semi-supervised low-variance generative models, diversity between base classifiers has to be structurally imposed.

2 Discriminative or Generative Models?

The generative/discriminative dilemma seems to divide Machine Learning into two separate communities.

In a statistical approach a classification problem is modelled by the joint distribution $\mathrm{p}(X, Y)$, where X and Y denote the data and the class random variables, respectively. Because we want to solve a classification problem, our goal is to find the optimal estimate of the class posterior $\mathrm{p}(Y|X)$. This can be determined via Bayes' rule:

$$\mathrm{p}(Y|X) = \frac{\mathrm{p}(X|Y)\mathrm{p}(Y)}{\mathrm{p}(X)}.$$

Discriminative classifiers directly model the class posterior distribution $\mathrm{p}(Y|X)$. In practical terms, this corresponds to modelling our problem as decision regions between classes. Typical example of discriminative models are neural networks, where we try to learn decision boundaries by minimising some error function. In a generative approach we make explicit assumptions about the form of the class conditional distributions $\mathrm{p}(X|Y)$ and class priors $\mathrm{p}(Y)$. Therefore, a generative model in practice models the data distribution rather than the decision regions. An example of a generative model is a Naïve Bayes network, which is based on the assumption that all the features are conditionally independent given the class:

$$\mathrm{p}(\boldsymbol{X}|Y) = \prod_{f=1}^{\mathrm{D}} \mathrm{p}(X_f|Y), \tag{1}$$

as depicted in Fig. 1. If all features are discrete, we can estimate Eq. (1) by frequency counts.

In terms of *problem applicability*, generative models can naturally incorporate unlabelled data because they learn the way data is distributed. On the other hand, discriminative models have no knowledge at all about the data distributions and therefore, their major drawback is that they cannot naturally handle unlabelled data. This implies that in semi-supervised problems there must exist an external mechanism that labels the unlabelled data before a discriminative classifier can incorporate them into the learning process, as in Co-training [3] and Tri-Training [4].

In terms of *performance*, when only few labelled data are available, there is strong evidence [9] that generative models outperform discriminative models.

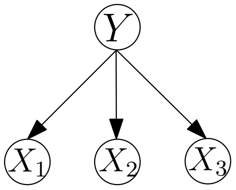


Fig. 1. A Naïve Bayes network. Each arc represent a inter-variable dependency, while the absence of an arc is an indication of independence between random variables.

Moreover, while discriminative models can achieve better performance, generative models have a faster speed of convergence. The main reason discriminative models are usually *preferred* to generative models is that the latter rely on strong assumptions. The choice of the *right* assumption in generative models is *crucial*, for studies have shown that when there is *model mismatch*, unlabelled data can degrade classification performance [10].

3 Semi-supervised Learning

3.1 Discriminative Ensembles for Semi-supervised Problems

The MCS community has traditionally focused on *discriminative base classifiers*, and most of the work done so far about semi-supervised learning has been concerned with techniques that let discriminative classifiers exploit unlabelled data [3,4,5,6]. As a *discriminative model cannot make use of data without labels, an external mechanism has to assign "pseudo-labels" to the unlabelled data before a classifier can effectively process them.* We now describe the main principles of how discriminative models can learn from unlabelled data. A full review of semi-supervised ensemble techniques is out of the scope of this paper; the reader might refer to [1,2] for a more extensive survey.

Decision-directed ensemble approaches [11] are based on the idea that a classifier can iteratively self-teach itself. Within this framework a single classifier is initially trained on the labelled data. Afterwards, the same classifier is used to classify unlabelled data and the most confident predicted patterns are selected and added along with their "pseudo label" to the labelled patterns; the process iterates until a stopping criterion is reached. In an ensemble approach the ensemble prediction could be used to assign pseudo labels to the unlabelled data. *Co-Training* [3] can be considered the first attempt to apply ensemble learning to semi-supervised problems. This approach is based on the assumption that the feature space can be split into two disjoint subsets called *views*, and that each one of these is sufficient for correct classification. Therefore, a single classifier is trained on each of these views. Initially, both classifiers are trained only on labelled data. Each classifier is then asked to classify a small amount of unlabelled data. The most confident predictions are added to the labelled training set of the other classifier; then the process re-iterates for a given amount of times. The basic idea behind co-training is that whenever classifiers disagree, the mistaken one can be "taught" by the other one, for each view is sufficient to make a correct prediction. In other words, co-training is an ensemble method that enforces *agreement on unlabelled data* [12]. An interesting extension is given by *tri-training* [4], where three classifiers use majority voting to label unlabelled data. If two classifiers agree, then the unlabelled pattern is labelled accordingly.

These "pseudo labels" are used only with discriminative classifiers. Although this mechanism might succeed, it seems somewhat ad-hoc. In contrast, generative models can lead to more elegant ensemble approaches.

3.2 Generative Ensembles for Semi-supervised Problems

The reason why we should be interested in generative models is that unlabelled data can be incorporated into their learning process without need of "pseudo labels". Once we have made our assumptions about the *form* of our joint distribution $\mathrm{p}(X, Y) = \mathrm{p}(X, Y, \theta)$, the learning process consists of finding the parameters θ that *most likely* fit our data[1].

For instance, let us consider a C class problem in a D dimensional space. In a semi-supervised problem, our data can be split into a finite set of labelled patterns $\mathcal{D}_{\mathrm{L}} = \{\boldsymbol{\mathcal{X}}_{\mathrm{L}}, \mathcal{Y}_{\mathrm{L}}\} = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, \mathrm{N}\}$ and a finite set of unlabelled data $\mathcal{D}_{\mathrm{U}} = \{\boldsymbol{\mathcal{X}}_{\mathrm{U}}\} = \{(\boldsymbol{x}_j) \mid j = \mathrm{N}+1, \ldots, \mathrm{M}\}$, $\mathcal{D} = \{\mathcal{D}_{\mathrm{L}}, \mathcal{D}_{\mathrm{U}}\}$. We assume that labelled and unlabelled patterns are independent and identically distributed samples drawn from the same joint probability distribution $\mathrm{p}(\boldsymbol{X}, Y)$. A semi-supervised Maximum Likelihood approach seeks to find the set of parameters θ that maximize the log-likelihood $\log \mathrm{p}(\mathcal{D}|\theta) = \log \mathrm{p}(\boldsymbol{\mathcal{X}}_L, \mathcal{Y}_L, \boldsymbol{\mathcal{X}}_U|\theta)$:

$$
\begin{aligned}
\log\ \mathrm{p}(\boldsymbol{\mathcal{X}}_L, \mathcal{Y}_L, \boldsymbol{\mathcal{X}}_U|\theta) = & \\
&= \log\ \mathrm{p}(\boldsymbol{\mathcal{X}}_L, \mathcal{Y}_L|\theta) + \log\ \mathrm{p}(\boldsymbol{\mathcal{X}}_U|\theta) \\
&= \sum_{i=1}^{N} \log \mathrm{p}(y_i|\theta) p(x_i|y_i, \theta) + \sum_{j=N+1}^{M} \log \sum_{k=1}^{C} \mathrm{p}(y_k|\theta) p(x_j|y_k, \theta)
\end{aligned}
\tag{2}
$$

From (2) it is easy to observe that the log-likelihood is made of two terms, the first one depending on the labelled data, and the second one depending on the unlabelled data. It follows that in a generative ensemble approach each base classifier can learn from labelled and unlabelled data, and in addition ensemble techniques could be used to improve classification accuracy.

4 Model Selection

Model selection is the process of choosing a specific class of models according to our knowledge of the problem. Whenever a generative model does not match the problem data distribution, we call this *model mismatch*.

[1] Alternatively we can use a full Bayesian Learning approach, which consists of integrating out parameters via approximation methods such as Variational Inference.

In generative models, we make assumptions about the *form* of probability distributions and about the *inter-variable dependencies* within these distributions. For instance we might assume that our data is Normally distributed, and we might also assume a Naïve Bayes approach, by making any feature conditioned on the class label statistically independent from any other, as depicted in Fig. 1. A model mismatch indicates our model does not represent the problem correctly because these independence assumption are violated in practice.

Similarly to generative models, discriminative classifiers are based on model assumptions, and therefore they are not always able to model boundaries between decision regions. A "model mismatch" in this case would correspond to selecting a linear perceptron to solve an XOR problem, or not using enough hidden nodes in our neural network.

The main difference between generative and discriminative approaches is that a mismatch is explicit for generative models, whereas it is hidden and more subtle for discriminative models: can the correspondence between the number of hidden nodes and decision boundaries be quantified in terms of model mismatch?

At a more abstract level, any learning algorithm can be thought of as a search in the space of *representable* models $\mathcal{H}$. The model mismatch problem then corresponds to asking the question: *What happens when the true model f does not belong to this search space $\mathcal{H}$?* This situation, which is depicted in Fig. 2, is known as the "representational problem" [13] in the MCS community.

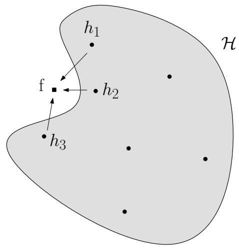


Fig. 2. An ensemble approach can deal with a representational problem by approximating the true hypothesis with a combination of wrong ones [13]

Discriminative ensemble learning tries to overcome this model limitation by replacing the single classifier approach with a combination of *accurate* and *different* models: if enough data are available, a combination of different models $h_1, h_2, \ldots, h_M$ in the search space can lead to a better approximation of the true model f even if this does not belong to $\mathcal{H}$ [13].

In theory the same ensemble principle could be applied to generative models. Moreover, we could exploit the property of generative models of *explicitly selecting* the model bias to *define* the boundaries of the hypothesis space. If the search space is then large enough, it might possible to combine *diverse* generative base models to achieve better performance than the single base classifier, and solve not only the representational problem but also the semi-supervised problem. We now illustrate some studies we have carried out on generative model ensembles.

5 Empirical Analysis

Very few experiments have been carried out on semi-supervised ensembles of generative models [7,8]. The aim of this study is to further investigate how unlabelled data can affect ensemble learning when we combine generative base classifiers.

The base model we chose for this analysis was a Gaussian Naïve Bayes network, i.e. a Naïve Bayes with Normally distributed continuous features and identity covariance matrix $\mathcal{N}(\boldsymbol{x}|\mu, \mathrm{I})$. We adopted a MAP approach and we used a scaled conjugate gradient descent algorithm to learn our model parameters. We applied three different ensemble methods: RSM [14] and two different variants of Bagging [14]: BaggingL– which samples with replacement from labelled data, and BaggingLU– which samples with replacement from labelled and unlabelled data. We used simple mean as a combination rule. Each technique has been evaluated according to a 5 times 2 statistical test. We tested our model on three different datasets, two of which were artificial datasets, the other was a real dataset:

Ringnorm. Artificial dataset that implements Breiman's ringnorm example. It is a 2 class problem with 20 features and it has 7400 patterns. This dataset is a *model mismatch* for our model, as one class has not been generated by $\mathcal{N}(\boldsymbol{x}|\mu, \mathrm{I})$.

Uniringnorm. Artificial dataset that represents a 2 class problem with 20 features and it has 1000 patterns. This dataset is a *model match* for our model, being the data generated from $\mathcal{N}(0, \mathrm{I})$ or $\mathcal{N}(\mu_2, \mathrm{I})$, where $\mu_2 = (a, a, ..., a)$, with $a = \frac{2}{\sqrt{20}}$.

Feltwell. We also applied our model on a real dataset by selecting 5124 patterns from Feltwell dataset. This is a 5 class problem with 15 features.

Following some experiments in [15], we studied how supervised and semi-supervised ensembles of generative models perform in comparison with the respective single classifier counterparts, as we increase the amount of labelled data. Our aim was to *identify any specific situation where semi-supervised ensemble learning is more beneficial than the semi-supervised single approach and the supervised ensemble.* Our results can be summarised as follow:

- *Data acts as a variance reducing factor.* Both semi-supervised ensembles and semi-supervised single classifiers show less variance than the supervised counterparts. This is unsurprising, as Naïve Bayes are low variance classifiers.
- BaggingLU
  - *Model match:* the semi-supervised ensemble performs exactly like the semi-supervised single classifier, and it always outperforms the supervised counterpart for any amount of labelled data.
  - *Model mismatch:* semi-supervised BaggingLU performs slightly worse than the semi-supervised single classifier, and in general semi-supervised learning outperforms the supervised one only when few labelled data are available (i.e. less than 40 labelled patterns).

– BaggingL
  - There is no difference between the semi-supervised ensemble and the semi-supervised single base classifier accuracy. *This implies that bagging the unlabelled data is effectively worsening the ensemble classification performance.*
– RSM
  - *Model match:* semi-supervised learning usually outperforms supervised learning for any amount of labelled data.
  - *Model mismatch:* In general semi-supervised learning outperforms supervised learning only when few labelled data are available (i.e. less than 50 labelled patterns). However, the ensemble techniques perform slightly and much (nearly 6%) worse, respectively, than the single counterparts for both supervised and semi-supervised learning.

We found similar results for Feltwell, where the semi-supervised ensemble techniques achieves almost the same accuracy as the semi-supervised respective single classifiers.

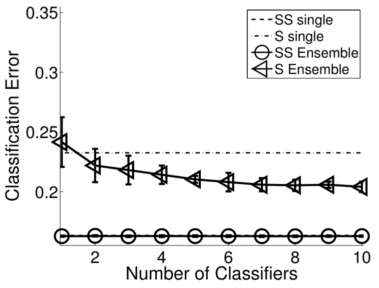
We conclude that Bagging and RSM techniques do not work well with semi-supervised low variance generative models, as data resampling or data random projections do not seem to increase the ensemble accuracy over the single classifier.

6 Discussion

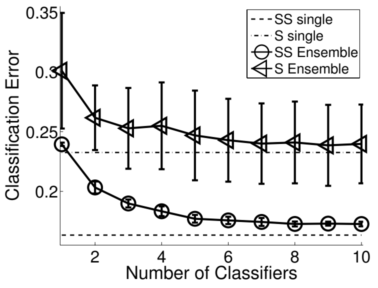
Both semi-supervised Bagging and RSM ensemble techniques seem not to improve classification accuracy over the single classifier approach – *but why?*

Let us focus on a typical semi-supervised scenario, where a large amount of unlabelled data and only few labelled data are available. We fix the amount of labelled data to be 30 patterns and we look at the ensemble behavior as we increase the number of base classifiers from 1 to 10. Results are shown in Figures 3 and 4 for a match problem and a mismatch problem, respectively. If we look at the leftmost part of both figures, we can observe the ensemble behavior of BaggingL as we increase the number of components in the ensemble. In both a model match and mismatch the semi-supervised ensemble error does not change as we increase the number of base classifiers, but at the same time this ensemble performs exactly like the semi-supervised single classifier. This is true for any amount of labelled data, and not only for 30 labelled patterns. A similar behavior has been observed for BaggingLU. It seems that when enough data are available, data resampling does not infer any kind of diversity on low variance generative base classifiers.

The rightmost part of both figures shows the ensembles created according to RSM. Whereas semi-supervised RSM fails for a model mismatch, Fig. 3 shows an unexpected behavior: *the semi-supervised ensemble error has very low variance and the error decreases as we increase the number of classifiers.* In other words, base classifiers are *diverse.* Increasing the amount of labelled data does not alter this behavior. However this semi-supervised ensemble does not perform better

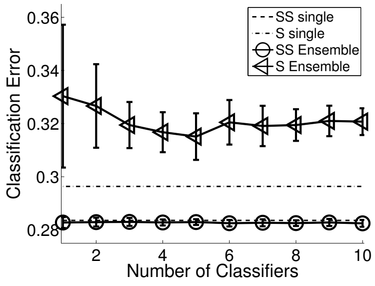


Model Match, BaggingL

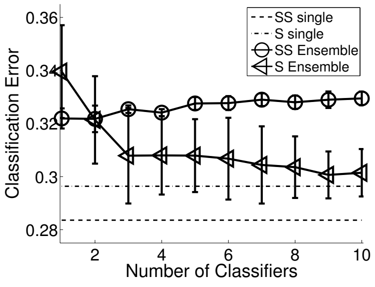


Model Match, RSM

Fig. 3. Classification error for a model match (Uniringnorm) with 30 labelled data. semi-supervised BaggingL does not create different base classifiers, whereas semi-supervised RSM does.



Model Mismatch, BaggingL



Model Mismatch, RSM

Fig. 4. Classification error for a model mismatch (Ringnorm) with 30 labelled data. semi-supervised BaggingL does not create different base classifiers, but the model mismatch cannot be model by the semi-supervised RSM.

than the semi-supervised single classifier. A possible reason for this could be that each base classifier is a projection of the feature space, and therefore it is missing information about the full data distribution, whereas the single classifier can model the data completely. From a representational problem perspective, this corresponds to the space of hypotheses being so small that it is not possible to find hypotheses that, if combined, can lead to a good approximation of the true function that represents our problem.

To sum up, our analysis of Bagging and RSM techniques shows how generative models cannot be learnt like their discriminative siblings:

– Resampling techniques like Bagging do not work well with low variance generative models, as the amount of training data acts like a variance reduction factor.
– RSM techniques introduce some diversity between base classifier components, but they do not outperform the single classifier. A reason for that

might be that the search space of the base classifier is not powerful enough to solve a representational problem.

Nevertheless this pattern of behavior looks promising and might indicate the need for novel generative ensemble techniques.

7 Conclusion and Future Work

How to Combine Generative Models?
While discriminative ensembles have the benefit of generating diverse base classifiers, base components require an external mechanism to make use of unlabelled data. *On the other hand, generative model ensembles naturally gain the ability of learning from unlabelled data but at the same time they lose in terms of diversity that can be generated by traditional ensemble techniques.*

Nevertheless, our results point towards the design of semi-supervised generative ensemble techniques that seek diversity in other ways than the traditional ones in MCS. It might be the case that generative model transparency can be exploited to build base classifiers that are *structurally diverse* and therefore extend the hypothesis search space. For instance, we could combine generative models that are characterised by different inter-model dependencies. An example is given by Super Parent One Dependency Estimator (SPODE) ensembles [16], where each base classifier feature depends not only on the class but also on another feature called superparent, as depicted in Fig. 5.

Generative models are the only *systematic* way we can explore hybrid ensembles because we can actually choose the structural difference between models. This is not possible with discriminative models, where it is not clear which boundaries might arise by combining different classifiers (for instance SVMs with neural networks). Instead, with generative models not only can we systematically place models in the search space but we can also decide how big the search space is.

A natural way to quantify diversity between generative models is given by the KL divergence [14], a non-commutative measure of the difference between probability distributions. Multivariate Mutual Information measures this KL divergence for multidimensional probability distributions. In practical terms it

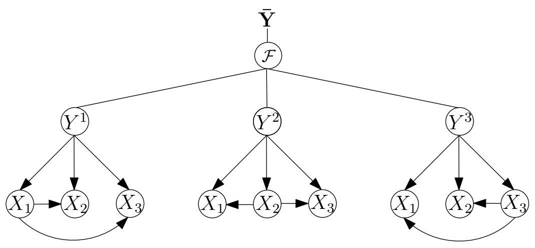


Fig. 5. An ensemble as a combination of all possible SPODEs

gives us an indication of how correlated, i.e. how diverse, random variables are [17]. The focus of our future work will be to use Multivariate Mutual Information to rank and select diverse generative base classifiers from the hypothesis space. This might allow us to solve both the representational and the semi-supervised problems in an ensemble fashion.

References

1. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison (2005)
2. Roli, F.: Semi-supervised multiple classifier systems: Background and research directions. In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS 2005. LNCS, vol. 3541, pp. 1–11. Springer, Heidelberg (2005)
3. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proc. of the 11th conf. on COLT, pp. 92–100 (1998)
4. Li, M.: Tri-training: Exploiting unlabeled data using three classifiers. IEEE Trans. on Knowledge and Data Engineering 17(11), 1529–1541 (2005)
5. Bennett, K.P., Demiriz, A., Maclin, R.: Exploiting unlabeled data in ensemble methods. In: Proc. of the 8th Int. Conf. on KDD, pp. 289–296 (2002)
6. Leskes, B.: The value of agreement, a new boosting algorithm. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS, vol. 3559, pp. 95–110. Springer, Heidelberg (2005)
7. Miller, D., Uyar, S.: A mixture of experts classifier with learning based on both labelled and unlabelled data. In: Proc. of Advances in NIPS, vol. 9, pp. 571–578 (1997)
8. Buc, F., Grandvalet, Y., Ambroise, C.: Semi-supervised marginboost. In: Proc. of Advances in NIPS, vol. 14 (2002)
9. Ng, A., Jordan, M.: On generative vs. discriminative classifiers: A comparison of logistic regression and naive bayes. In: Proc. of Advances in NIPS, vol. 15 (2002)
10. Cozman, F.G., Cohen, I., Cirelo, M.C., et al.: Semi-supervised learning of mixture models. In: 20th International Conference on Machine Learning, pp. 99–106 (2003)
11. Martínez, C., Fuentes, O.: Face Recognition Using Unlabeled Data. Computación y Sistemas 7(2), 123–129
12. Balcan, M., Blum, A.: An Augmented PAC Model for Semi-Supervised Learning. In: Chapelle, O., et al. (eds.) Semi-Supervised Learning. The MIT Press, Cambridge (2006)
13. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
14. Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms. Wiley Press, Chichester (2004)
15. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.M.: Text classification from labeled and unlabeled documents using EM. Machine Learning 39(2/3) (2000)
16. Yang, Y., Webb, G., et al.: To select or to weigh: A comparative study of model selection and model weighing for spode ensembles. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS, vol. 4212, pp. 533–544. Springer, Heidelberg (2006)
17. Brown, G.: A new perspective on information theoretic feature ranking. In: 12th Int. Conf. on Artificial Intelligence and Statistics (2009) (to appear)

Hierarchical Ensemble Support Cluster Machine

Mingmin Chi[1], Youdong Miao[1], Youze Tang[1], Jón Atli Benediktsson[2], and Xuanjing Huang[1]

[1] School of Computer Science, Fudan University, Shanghai, China
{mmchi,082024068,07300720181,xjhuang}@fudan.edu.cn
[2] Faculty of Electrical and Computer Engineering, University of Iceland, Iceland
benedikt@hi.is

Abstract. In real applications, a large-scale data set is usually available for a classifier design. The recently proposed Support Cluster Machine (SCM) can deal with such a problem, where data representation is firstly changed with a mixture model such that the classifier works on a component level instead of individual data points. However, it is difficult to decide the proper number of components for designing a successful SCM classifier. In the paper, a hierarchical ensemble SCM (HESCM) is proposed to address the problem. Initially, a hierarchical mixture modeling strategy is used to obtain different levels of mixture models from fine representation to coarse representation. Then, the mixture model in each level is exploited for training SCM. Finally, the learnt models from all the levels are integrated to obtain an ensemble result. Experiments carried on two real large-scale data sets validate the effectiveness of the proposed approach, increasing classification accuracy and stability as well as significantly reducing computational and spatial complexities of a supervised classifier compared to the state-of-the-art classifiers.

1 Introduction

Support Vector Machines (SVMs) have been successfully applied to many applications, such as text categorization [4], face detection [7], remote sensing [6]. However, it is difficult to handle large-scale data sets. Recently, a SVM-like kernel based approach, the Support Cluster Machine (SCM) [5], was proposed to address the problem in the machine learning community. The main idea in the SCM is that the representation of data is changed with a mixture model. Then, a similarity measure between generative models is defined by a kernel, i.e., probability product kernel (PPK) [3]. Unlike the SVM, the classifier is trained in a probability space where the learnt models contain support *clusters* rather than support vectors (the name SCM is based on this).

In the SCM, it is important to obtain a proper number of mixture models for the best classification result. If the selected number of mixture models does not fit the data well, the classification accuracy can decrease. In large-scale classification problems, it is time-consuming to select the proper number of mixture components by model selection. To address this problem, the data are here represented in a hierarchical structure with mixture modeling such that different descriptions of data can be captured. Then, the SCM is trained by using the mixture model from each level to avoid an improper statistical information. Finally, the classification result is integrated by using the results provided by different SCM classifiers that are designed using mixture models at different levels. Note that the data are represented in a hierarchical structure with different granularity by going from many (fine representation) to few (coarse representation) numbers of models. Therefore, the diversity of data is obtained in the input space to a base classifier. Experiments based on two real large-scale data sets validate the effectiveness of the proposed approach, which improves the classification accuracy and reliability as well as significantly reduces the computational complexity (linear scalability) when compared to the-state-of-the-art classifiers.

The rest of the paper is organized as follows. The next section describes the proposed hierarchical ensemble SCM. Section 3 discusses the data used in the experiments, and reports and discusses the results provided by the different algorithms. Finally, conclusions and discussion are given in Section 4.

2 Proposed Approach

Let the given training data set $\mathrm{X}_l = (\mathbf{x}_i, y_i)_{i=1}^{n}, \mathrm{X}_l \in \mathcal{R}^{D\times n}$ be made up of n labeled samples in a D-dimensional input space. Without a loss of generality, we work on a binary classification problem, i.e., $y_i = +1$ if $\mathbf{x}_i$ is labeled as the positive class and -1 otherwise.

In the proposed algorithm, data are represented by a mixture model, e.g.,a Mixture of Gaussians (MoG) (not limited to MoG) which contains a local information. Therefore, the learning procedure is done in a probability space instead of input feature space and only learnt components rather than data points are used for training. That is the reason why the proposed approach is more scalable than the common SVM trained by data points. In our framework, a Support Cluster Machine is used for training to satisfy the above condition. However, it is difficult to evaluate the influence of the number of mixtures on the classification results [5]. Therefore, different number of mixtures are modeled in a hierarchy and used as inputs to base classifiers, i.e., SCMs. Finally, a classification result is integrated in terms of those provided by base classifiers to improve classification accuracy and stability.

2.1 Hierarchical Mixture Modeling

In real applications, it is difficult to identify a suitable number of components for the design of a SCM classification task. In the paper, a hierarchical representation of data is adopted such that different numbers of mixture models can be obtained. For a large-scale data problem, we need a computationally efficient mixture hierarchical clustering algorithm for the scalability. In [9], an Expectation Maximization (EM)-like [2] clustering algorithm was used to satisfy this condition which was proposed to construct mixture hierarchies in terms of a finite set of virtual samples in a bottom-up fashion. Here, all subsequent computations are based on the representatives in the previous level and the original points no longer need to be considered. Namely, the learning of a given level fully exploits the previous computation.

As in commonly known mixture modeling, the datum at a given level is expressed in the form:

$$P(\mathrm{X}) = \sum_{i=1}^{K^l} \pi_i^l P(\mathrm{X}|z_i^l = 1, \mathcal{M}_l), \tag{1}$$

where l is the level in the hierarchy, $\mathcal{M}^l$ the mixture model at this level, K^l the number of components for the model, π_i^l the prior probability of the i^{th} component, and z_i^l a binary variable that takes the value 1 if and only if the sample X was drawn from the i^{th} component.

The basic problem for the hierarchial mixture modeling is how to estimate the parameters for the l^{th} level given those in the $(l+1)^{\text{th}}$ one. As in a flat mixture model, let us assume that the virtual samples are independent such that the likelihood of the virtual samples under the model $\mathcal{M}_l$ can be written as follows:

$$P(\mathrm{X}|\mathcal{M}_l) = \prod_{i=1}^{K^{l+1}} P(\mathrm{X}_i|\mathcal{M}_l).$$

and with the constraint that samples in the same block are assigned to the same component of $\mathcal{M}_l$, i.e.,

$$P(z_i^{l+1} = 1) = P(z_j^l = 1|z_i^{l+1} = 1)P(z_j^l = 1). \tag{2}$$

This means that the node i of level $l+1$ is a child of the node j of level l.

Therefore, the likelihood of the complete visual data in level l can be given by

$$P(\mathrm{X}, Z|\mathcal{M}_l) = \prod_{i=1}^{K^{l+1}} \prod_{j=1}^{K^l} \left[\pi_j^l P(\mathrm{X}_i|z_{ij} = 1, \mathcal{M}_l)\right]^{z_{ij}}$$

where $z_{ij} = z_i^{l+1} z_j^l$ is a binary variable that takes 1 if and only if the block (virtual sample) X_i is assigned to the j^{th} component of $\mathcal{M}_l$, and the real sample $\mathbf{x}_i^m$ is the m^{th} data point in X_i. Using log-likelihood, we have the following Expectation step:

$$h_{ij} = \mathrm{E}[z_{ij}|\mathrm{X}_i, \mathcal{M}_l] = P(z_{ij} = 1|\mathrm{X}_i, \mathcal{M}_l) = \frac{P(\mathrm{X}_i|z_{ij} = 1, \mathcal{M}_l)\pi_j^l}{\sum_k P(\mathrm{X}_i|z_{ik} = 1, \mathcal{M}_l)\pi_k^l}.$$

For the Gaussian case, h_{ij} can be computed by

$$h_{ij} = \frac{\left[\frac{1}{(2\pi)^{(\frac{d}{2})} \prod_{k=1}^d \sigma_{ij}^l} \exp\left(-\frac{1}{2}(\sum_{k=1}^d [\frac{(\mu_{ij}^l - \mu_{ij}^{l+1})^2}{(\sigma_{ij}^l)^2} + \frac{(\sigma_{ij}^{l+1})^2}{(\sigma_{ij}^l)^2}])\right)\right]^{M_i} \pi_j^l}{\Sigma_k \text{numerator, with } k \text{ in place of } j}. \tag{3}$$

The Maximization-step consists of maximizing

$$Q^l = \sum_{i=1}^{K^{l+1}} \sum_{j=1}^{K^l} h_{ij} \log\left(\pi_j^l Pr(\mathrm{X}_i | z_{ij}, \mathcal{M}_l)\right) \text{ s.t. } \sum_j \pi_j^l = 1. \tag{4}$$

For the Gaussian case, we can easily obtain the model parameters (π, μ, Σ) as,

$$\pi_j^k = \frac{\sum_i h_{ij}}{\mathrm{C}^{l+1}} \tag{5}$$

$$\mu_j^l = \frac{\sum_i h_{ij} M_i \mu_i^{l+1}}{\sum_i h_{ij} M_i} \tag{6}$$

$$\Sigma_j^l = \frac{1}{\sum_i h_{ij} M_i}\left[\sum_i h_{ij} M_i \Sigma_i^{l+1} + \sum_i h_{ij} M_i (\mu_i^{l+1} - \mu_j^l)(\mu_i^{l+1} - \mu_j^l)^\top\right]. \tag{7}$$

2.2 Support Cluster Machine (SCM)

After obtaining the mixture of Gaussians, the similarity measure between Gaussians is defined and a SVM-like learning framework is adopted for learning. After training, the kernel between a Gaussian and a vector is also defined for prediction.

Learning phase. Unlike the SVM, the SCM maximizes the margin between the positive and negative `clusters`, rather than data vectors, i.e.,

$$\min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C \sum_{k=1}^{K} \pi_k \xi_k \tag{8}$$

with the constraints

$$y_k \left(\mathbf{w}^\top \phi(\boldsymbol{\theta}_k) + b\right) \geq 1 - \xi_k, \quad k = 1, \ldots, K \tag{9}$$

where $\phi(\cdot)$ is a mapping function and the slack ξ_k is multiplied by the weight π_k (the prior of the k^{th} cluster in MoG) such that a misclassified cluster with more samples can be given a heavier penalty [5].

Incorporating the constraints (9) and $\xi_k \geq 0, k = 1, \ldots, K$, to the cost function (8), and using Lagrangian theorem, the constrained optimization problem can be transformed into a dual problem following the same step as that in SVM [8]. Thus, the dual representation of SCM is given as

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{K} \alpha_k - \frac{1}{2} \sum_{k=1}^{K} \sum_{k'=1}^{K} y_k y_{k'} \alpha_k \alpha_{k'} \boldsymbol{\kappa}(\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k'}) \tag{10}$$
$$\text{s.t.} \begin{cases} 0 \leq \alpha_k \leq \pi_k C, & k = 1, \ldots, K \\ \sum_{k=1}^{K} \alpha_k y_k = 0. \end{cases}$$

The SCM has the same optimization formulation as the SVM except that in the SCM the Lagrange multipliers α_k are bounded by C *multiplied by the weight* π_k.

Kernel Function by Generative Models. After preprocessing, the data are represented by a mixture of Gaussians. The similarity between the components can be calculated by the probability product kernel (PPK) [3]:

$$\begin{aligned}\boldsymbol{\kappa}_{kk'} &= \boldsymbol{\kappa}(\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k'}) \\ &= (\pi_k \pi_{k'})^\rho \int_{\mathcal{R}^D} \mathcal{N}^\rho(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\mathcal{N}^\rho(\mathbf{x}|\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})d\mathbf{x} \\ &= (\pi_k \pi_{k'})^\rho \rho^{-D/2}(2\pi)^{\frac{(1-2\rho)D}{2}} |\widetilde{\boldsymbol{\Sigma}}|^{-\frac{1}{2}} |\boldsymbol{\Sigma}_k|^{-\frac{\rho}{2}} |\boldsymbol{\Sigma}_{k'}|^{-\frac{\rho}{2}} \\ &\exp\big(-\frac{\rho}{2}(\boldsymbol{\mu}_p^\top \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\mu}_k + \boldsymbol{\mu}_{k'}^\top \boldsymbol{\Sigma}_{k'}^{-1} \boldsymbol{\mu}_{k'} - \widetilde{\boldsymbol{\mu}}^\top \widetilde{\boldsymbol{\Sigma}}^{-1} \widetilde{\boldsymbol{\mu}})\big)\end{aligned} \tag{11}$$

where ρ is a constant, $\widetilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_k^{-1} + \boldsymbol{\Sigma}_{k'}^{-1}$, $\widetilde{\boldsymbol{\mu}} = \boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k + \boldsymbol{\Sigma}_{k'}^{-1}\boldsymbol{\mu}_{k'}$.

To reduce the computational cost, it is assumed that the features are statistically independent. Hence, a diagonal covariance matrix is used, i.e., $\boldsymbol{\Sigma}_k = \text{diag}((\sigma_k^{(1)})^2, \cdots, (\sigma_k^{(d)})^2)$. Therefore, we have the PPK as

$$\begin{aligned}\boldsymbol{\kappa}_{kk'} &= (\pi_k \pi_{k'})^\rho \rho^{-D/2}(2\pi)^{\frac{(1-2\rho)D}{2}} \prod_{d=1}^{D} \frac{(\sigma_k^{(d)}\sigma_{k'}^{(d)})^{(1-\rho)}}{\sqrt{(\sigma_i^{(d)})^2 + (\sigma_j^{(d)})^2}} \\ &\exp\left(-\frac{\rho}{2} \sum_{d=1}^{D} \frac{(\mu_k^{(d)} - \mu_{k'}^{(d)})^2}{(\sigma_k^{(d)})^2 + (\sigma_{k'}^{(d)})^2} \right).\end{aligned} \tag{12}$$

Prediction (classification of unlabeled samples). A test sample $\mathbf{x}$ can be treated as an extreme case of Gaussian $\boldsymbol{\theta}_\mathbf{x}$ when its covariance matrix vanishes, i.e., $\boldsymbol{\theta}_\mathbf{x} = (\pi_\mathbf{x} = 1, \boldsymbol{\mu}_\mathbf{x} = \mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x} = \sigma_\mathbf{x}^2\mathbf{I}, \sigma_\mathbf{x} \propto 0)$, where $\mathbf{I}$ is an identity matrix.

The kernel value between $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}_\mathbf{x}$ is computed using (12). When $\rho = 1$, we get the kernel value for the SCM prediction:

$$\begin{aligned}\kappa(\boldsymbol{\theta}_k, \boldsymbol{\theta}_\mathbf{x}) &= \pi_k \frac{1}{(2\pi)^{D/2} \det|\Sigma_k|} \exp\left(-\sum_{d=1}^{D} \frac{(\mu_k^{(d)} - x^{(d)})^2}{2(\sigma_k^{(d)})^2} \right) \\ &= \pi_k p(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\end{aligned} \tag{13}$$

which is the posterior probability of $\mathbf{x}$ given $\boldsymbol{\theta}_k$.

Then, the prediction function is the linear combination of the kernels computed between trained mixture components and the test pattern $\boldsymbol{\theta}_\mathbf{x} = \{1, \mathbf{x}, \sigma_x^2\mathbf{I}\}$ as follows:

$$f(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k y_k \boldsymbol{\kappa}(\boldsymbol{\theta}_k, \boldsymbol{\theta}_\mathbf{x}) + b. \tag{14}$$

Accordingly, a class label is assigned to a test pattern by

$$\mathbf{x} \in \begin{cases} +1, & \text{if } f(\mathbf{x}) \geq 0 \\ -1, & \text{otherwise} \end{cases} = \text{sgn}(f(\mathbf{x})). \tag{15}$$

2.3 Hierarchical Ensemble SCM (HESCM)

After the estimation of the model parameters in the hierarchy, a serial of sets of model parameters $\boldsymbol{\Theta}^l, l = 1, \cdots, L$ for different levels are obtained. In terms of the mixture model for a given level, we can train a support cluster machine as described in Section 2.2. Then, the classification result is integrated in terms of those provided by the individual classifiers.

For a supervised classification task, a labeling information is known and consequently a hierarchical mixture model is estimated within the class. Accordingly, inputs to different SCMs are $\{\boldsymbol{\Theta}^l, \mathbf{y}^l\}, l = 1, \cdots, L$, where $\mathbf{y}^l$ is the corresponding class vector for $\boldsymbol{\Theta}^l$ in the l^{th} level, and L is the number of levels and also the number of base classifiers. The prediction function for each classifier f^l is a linear combination of the kernels computed between the learnt mixture component and a test pattern $\boldsymbol{\theta}_{\mathbf{x}}$:

$$f^l(\mathbf{x}) = \sum_{k=1}^{K^l} \alpha_k^l y_k^l \boldsymbol{\kappa}(\boldsymbol{\theta}_k^l, \boldsymbol{\theta}_{\mathbf{x}}) + b^l. \tag{16}$$

Then, for the l^{th} base classifier, a class label is assigned to the test pattern in the form

$$\mathbf{x} \in \operatorname{sgn}\left(f^l(\mathbf{x})\right). \tag{17}$$

Finally, a winner-takes-all combination rule is used to make a final decision [1],

$$\mathbf{x} \in y_m \text{ if } y_m = \operatorname{argmax}_c N_c, \sum_c N_c = L \tag{18}$$

where N_c is the accumulated number that the base classifiers assign the y_c labeling to the test pattern.

3 Experimental Results

In this section we present experimental results that demonstrate the effectiveness of the proposed HESCM by comparison to the Hierarchical Ensemble Representative Approach (HERA), the Representative Approach (RA), the SCM and the SVM. Regarding the RA, it simply keeps the mean vectors $\boldsymbol{\mu}_k$ of mixture model as representatives. Then, mean vectors instead of original data points are used to train a standard SVM. Similar to the proposed HESCM, the HERA integrates the results provided by the RAs from different hierarchical levels.

3.1 Experimental Setting

In the experiments, the SVM and the RA used a Gaussian kernel with the kernel width parameter σ. For the model selection, there is an additional parameter, i.e., the penalization parameter C. In the proposed HESCM, C is fixed to 100, and the number of base classifiers L and the number of components K^L in the bottom level should be first decided. Regarding HERA, except for two parameters L and K^L, the same setting as HESCM, there are two other parameters, σ, and C which should be selected during model selection.

We tested the algorithms with two real large-scale benchmark machine learning data sets:

`Adult`[1]: 1994 Census data base, where there are 30, 162 training samples and 15, 060 test samples with 14 features, and the percentage of positive samples is 24.78%. Prediction task is to determine whether a person makes over $ 50K a year.

`IJCNN1`[2]: Contains 49,990 (4,853 positive and 45,137 negative) training samples and 91,701 (8,712 positive and 82,989 negative) test examples with 22 attributes.

3.2 Experimental Results

Adult Data. For this data set, $L = 15$ hierarchical mixture models were obtained in a hierarchical mode. Therefore, there are 15 base classifiers in total. From the bottom up, 90% components of the $(l+1)^{\text{th}}$ level were kept for the l^{th} level. For instance, at the last level (i.e., 15^{th}) level, there are $K^{15} = 1000$ components, and, therefore, $K^{14} = 900$ components. The accuracies of individual classifiers are shown in Table 1 using the RA and the SCM. From the analysis of Table 1, one can see that in most of cases, classification accuracies provided by the SCMs trained by all the model information are better than those provided by the RAs trained by the mean vectors. This is possibly caused by the fact that the SCM takes advantage of more information including the mean vectors $\boldsymbol{\mu}_k$ as

Table 1. Classification accuracies at individual levels using the RA and the SCM and the number of mixture components (or vectors), K^l, $l = 1, \cdots, 15$ for the Adult data set

Level	K^l	RA (%)	SCM (%)
1	228	75.43	74.13
2	253	75.43	74.07
3	281	75.43	74.54
4	313	75.57	74.95
5	348	75.84	75.14
6	387	75.91	74.96
7	430	76.00	75.73
8	478	76.43	77.35
9	531	76.85	77.37
10	590	76.67	78.42
11	656	77.17	78.49
12	729	77.66	77.90
13	810	77.90	78.42
14	900	78.15	79.66
15	1000	78.51	80.06

[1] Available at: http://archive.ics.uci.edu/ml/datasets/Adult
[2] Available at: http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/binary.html

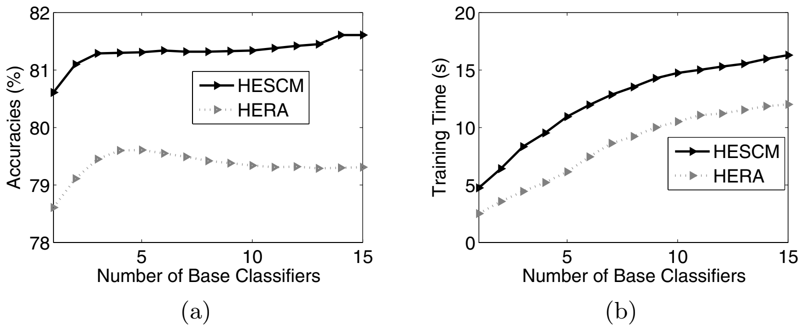


Fig. 1. The ensemble classification accuracies and the corresponding training time with respect to the number of base classifiers for the Adult data set

well as component proportion π_k and covariance matrices $\boldsymbol{\Sigma}_k$ to compute the similarity of the generative models when compared to the RA which uses only mean vectors.

Although the accuracies of the individual classifiers decreased from the bottom level to the higher levels, all the ensemble accuracies using different number of base classifiers were higher than the one provided by the single SCM shown in Fig. 1(a) and also higher than those provided by individual SCMs. Moreover, the classification accuracies increase smoothly with the increase in the number of base classifiers. This is mainly due to the diversity of the results provided by the mixture models in hierarchy. However, the training time only increase slightly as shown in Fig. 1(b). Furthermore, it is important to note that the HESCM is computationally much less demanding than the standard SVM, e.g., 172.23s[3] for Adult data set using libsvm[4], while the accuracy (83.73%) is only slightly better than the proposed HESCM. By the analysis of Fig. 1(a), one can see that the accuracies by HESCMs are also much better than those by HERAs.

IJCNN1 Data. Similar to the experiments on Adult data, we obtained the classification accuracies at individual levels using the RA and the SCM on the IJCNN1 data set shown in Table 2. For this data set, we utilized a six-level hierarchical structure and used the coarser granulity to decrease the number of kernels at each level. Accordingly, the training times decreases significantly (cf. Fig. 2(b)). However, the ensemble accuracies are improved steadily with the increase of base classifiers shown in Fig. 2(a). Again, one can see from that all the ensemble accuracies are better than the initial one, 94.30% by the SCM, and the others provided by a single classifier. Note that, the SCM obtains much better classification accuracies than the SVM in terms of representative vectors. This further confirms the effectiveness of the model-based kernels for the classification tasks.

[3] If the time for model selection is also considered, the computational complexities are much demanding.

[4] Available at: http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

Table 2. Classification accuracies at individual levels using the RA and the SCM and the number of mixture components (or vectors), K^l, $l = 1, \cdots, 6$ for IJCNN1 data set

Level	K^l	RA (%)	SCM (%)
1	18	76.79	88.88
2	37	78.18	83.05
3	75	82.68	88.39
4	150	84.83	89.51
5	300	84.55	92.76
6	600	90.49	94.30

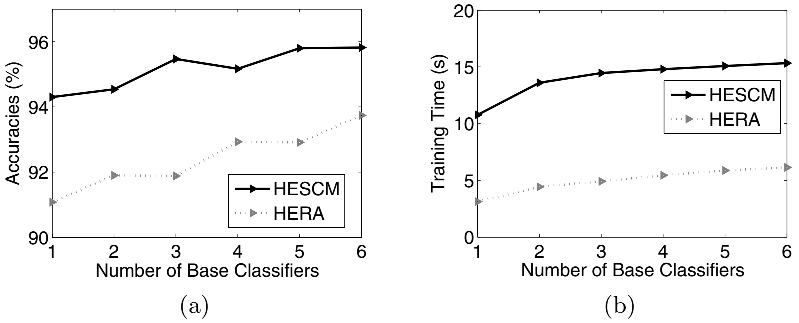


Fig. 2. The ensemble classification accuracies and the corresponding training time with respect to the number of base classifiers on IJCNN1 data set

4 Discussion and Conclusion

In the paper, we propose a hierarchical ensemble classification algorithm to integrate results provided by support cluster machines. In particular, a series of mixture models in a hierarchical manner were generated to define SCM base classifiers. In each level, it is not necessary to scan the original data base to estimate mixture models, while visual samples in the current generated in terms of those in the previous bottom level are generated for the estimation of mixture models. Therefore, the computational complexity decreases from the bottom level to higher levels. Although the statistical modeling of mixture models loses some information from the bottom to higher levels, the diversity of mixture models for defining base classifiers still increases the ensemble accuracy as a whole. This can be validated by the experiments shown in Section 3 on two large real data sets, which results in better classification accuracies compared to the SCM, the RA, and the HERA and also needs a much less computation time than the SVM.

In the proposed HESCM algorithm, data are represented by mixture models using a clustering technique. The approach can suffer from the curse of dimensionality problem. A dimensionality reduction technique, such as random projection, together with a hierarchical mixture modeling approach will be developed for addressing that problem in our future research.

Acknowledgements

This work was supported in part by Natural Science Foundation of China under contracts 60705008 and 60673038, Ph.D. Programs Foundation of MoE of China under contract 20070246132 and the Research Fund of the University of Iceland.

References

1. Briem, G.J., Benediktsson, J.A., Sveinsson, J.R.: Multiple classifiers in classification of multisource remote sensing data. IEEE Trans. Geosci Remote Sensing 40(10), 2291–2299 (2002)
2. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. The Royal Statistical Society, Series B (1977)
3. Jebara, T., Kondor, R., Howard, A.: Probability product kernels. Journal of Machine Learning Research 5, 819–844 (2004)
4. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Springer, Heidelberg (1997)
5. Li, B., Chi, M., Fan, J., Xue, X.: Support cluster machine. In: Proceedings of the 24th International Conference on Machine Learning, Corvallis, USA, June 2007, pp. 505–512 (2007)
6. Melgani, F., Bruzzone, L.: Classification of hyperspectral remote sensing images with support vector machines. IEEE Trans. Geosci. Remote Sensing 42(8), 1778–1790 (2004)
7. Osuna, E., Freund, R., Girosit, F.: Training support vector machines: an application to face detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 1997), June 1997, pp. 130–136 (1997)
8. Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press, Cambridge (2002)
9. Vasconcelos, N., Lippman, A.: Learning mixture hierarchies. In: Proceedings of the 1998 conference on Advances in neural information processing systems II, pp. 606–612. MIT Press, Cambridge (1999)

Multi-scale Stacked Sequential Learning

Oriol Pujol[1,2], Eloi Puertas[1], and Carlo Gatta[2]

[1] Dept. Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Gran Via 585, 08007, Barcelona, Spain
[2] Computer Vision Center, Campus UAB, Edifici O, 08193, Bellaterra, Barcelona, Spain

Abstract. One of the most widely used assumptions in supervised learning is that data is independent and identically distributed. This assumption does not hold true in many real cases. Sequential learning is the discipline of machine learning that deals with dependent data such that neighboring examples exhibit some kind of relationship. In the literature, there are different approaches that try to capture and exploit this correlation, by means of different methodologies. In this paper we focus on meta-learning strategies and, in particular, the stacked sequential learning approach. The main contribution of this work is two-fold: first, we generalize the stacked sequential learning. This generalization reflects the key role of neighboring interactions modeling. Second, we propose an effective and efficient way of capturing and exploiting sequential correlations that takes into account long-range interactions by means of a multi-scale pyramidal decomposition of the predicted labels. Additionally, this new method subsumes the standard stacked sequential learning approach. We tested the proposed method on two different classification tasks: text lines classification in a FAQ data set and image classification. Results on these tasks clearly show that our approach outperforms the standard stacked sequential learning. Moreover, we show that the proposed method allows to control the trade-off between the detail and the desired range of the interactions.

1 Introduction

As the machine learning community matures, problems it addresses become more challenging. One of the most widely used assumptions in supervised learning is that data is independent and identically distributed (iid). However, there are many real world applications in which that assumption does not necessarily hold. Consider the case of a laughter detection application from voice records. Laugh has a clear pattern alternating voice and non-voice segments. Thus, discriminant information comes from the alternating pattern, and not just by the samples on their own. Another case is part-of-speech tagging in which each example describes a word that is categorized as noun, verb, adjective, etc. In this case it is very unlikely that patterns such as [verb, verb, adjective, verb] occur. All these applications present a common feature: the sequence/context of the labels matters.

Sequential learning [1] breaks the iid assumption and assumes that samples are not independently drawn from a joint distribution of the data samples $\mathbf{X}$ and their labels Y. In sequential learning the training data actually consists of sequences of pairs $(\mathbf{x}, y)$, so that neighboring examples exhibit some kind of correlation. Usually sequential learning applications consider one-dimensional relationship support, but this kind of relationships appear very frequently in other domains, such as images, or video. Consider the case of object recognition in image understanding. It is clear that if one pixel belongs to a certain object category, it is very likely that neighboring pixels also belong to the same object (with the exception of its borders).

Sequential learning is often confused with a very related application, time series prediction. The main difference between both problems lays in the fact that sequential learning has access to the whole data set before any prediction is made and the full set of labels is to be provided at the same time. On the other hand, time series prediction has access to real labels up to the current time t and the goal is to predict the label at $t + 1$.

In literature, sequential learning has been addressed from different perspectives: from the point of view of meta-learning by means of sliding window techniques, recurrent sliding windows [1] or stacked sequential learning [5] [2], the method is formulated as combination of classifiers. From the point of view of graphical models, using Hidden Markov Models, Conditional Random Fields [6] to infer the joint or conditional probability of the sequence. And Graph Transformer Networks [9], that considers the input and output as a graph and looks for the transformation than minimizes a loss function of the training data using a Neural Network.

Independently of the specific method, there are still fundamental issues in sequential supervised learning that requires the attention of the community. In [1] the author acknowledge the following ones: a) How to capture and exploit sequential correlations; b) how to represent and incorporate complex loss functions; c) how to identify long-distance interactions; d) how to make sequential learning computationally efficient.

In this work, we are concerned with meta-learning strategies. Recently, Cohen et al. [2] showed that stacked sequential learning (SSL from now on) performed better than CRF and HMM on a subset of problems called "sequential partitioning problems". These problems are characterized by long runs of identical labels. Moreover, SSL is computationally very efficient since it only needs to train two classifiers a constant number of times. Considering these benefits, we decide to explore in depth sequential learning using SSL and generalize the Cohen architecture to deal with a wider variety of problems as well as giving answers to most of the open problems described by Dietterich in [1].

In this paper, we argue that a fundamental and overlooked step in SSL is the way the extended set is created. We first provide a general framework in which this extension step is clearly identified and then propose a new aggregation method capable of capturing long-distance interactions efficiently. The proposed step is based on a multi-scale decomposition [7] of the predicted data labels.

As a result, we provide answers to the different open issues obtaining a method that a) captures and exploit sequential correlations b) since the method is a meta-learning strategy the loss function dependency is delegated to the second step classifier; c) it efficiently captures long-distance interactions; and d) it is fast, because it relies on training a few general learners. The benefits of the new method are shown in a one-dimensional support problem in a FAQ structure detection dataset. Moreover, along with one-dimensional sequence examples, we provide results and discussion in the image domain using 2-D support – note that image processing and understanding is a good example of *sequential partitioning problems*. For the 2D domain, we use the Weizmann horse database [4].

2 Multiscale Stacked Sequential Learning

The basic idea of stacked sequential learning is to create an extended data set that joins the original training data features with the predicted labels considering a neighborhood around the example. The basic SSL method uses a five-fold cross-validation on the training set to obtain the predicted set Y' and considers a sliding window of length w with origin in the prediction of the current example to extend its features. That is for each example in the training set $x_i \in \mathbf{X}$, the predicted values $y'_i \in Y'$ are obtained and joined creating an extended example $x_i^{ext} = (x_i, y'_{i-w_a}, \dots, y'_{i+w_b}) \in \mathbf{X^{ext}}$, where the number of added features is $w = w_a + w_b + 1$.

In our opinion, the core step in this process is the creation of the extended data set. That is, how the neighboring interactions are captured. In the original SSL article [2] they propose to use a sliding window. However, this is a very poor choice when one want to consider far interaction patterns or when one needs to deal with noisy environments, since the training becomes very sensitive to errors in the neighborhood. For this reason, in the next subsection, we extend the basic model to cope with a general neighboring definition and then propose the use of a multi-scale approach that includes the basic windowing strategy as a particular case. Additionally, it allows to deal with long distance interactions as well as to use a priori knowledge for modeling the neighboring interactions.

2.1 Generalized Stacked Sequential Learning

The framework for generalizing the stacked sequential learning includes a new block in the pipeline of the basic SSL. Figure 1 shows the Generalized Stacked Sequential Learning process. A classifier $h_1(x)$ is trained with the input data set $(\mathbf{x}, y)$ by means of cross-validation and the predicted labels y' are obtained. The next block defines the policy for creating the neighborhood model of the predicted labels. $z = J(y', \rho, \theta) : \mathcal{R} \rightarrow \mathcal{R}^w$ is a function that captures the data interaction with a model parameterized by θ in a neighborhood ρ. The result of this function is a w-dimensional value, where w is the number of elements in the support lattice of the neighborhood ρ. In the case of defining the neighborhood by means of a window, w is the number of elements in the window. Then, the

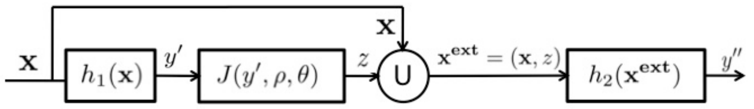


Fig. 1. Block diagram for the generalized stacked sequential learning

output of $J(y',\rho,\theta)$ is joined with the original training data creating the extended training set $(\mathbf{x^{ext}}, y) = ((\mathbf{x}, z), y)$. This new set is used to train a second classifier $h_2(\mathbf{x^{ext}})$ with the goal of producing the final prediction y''. Observe, that the system will be able to deal with neighboring relations depending on how well one is able to characterize them in $J(y',\rho,\theta)$.

2.2 Multiscale Stacked Sequential Learning (MSSL)

The generalized stacked sequential learning emphasizes the key role of the interaction modeling by means of $J(y',\rho,\theta)$. The proposed definition of $J(y',\rho,\theta)$ consists of two steps that answers two fundamental questions: first, how to model the relationship between neighboring locations and second, how to define the support lattice to produce the final set z. To answer the first question, we propose the use of a multi-scale approach by means of a pyramidal decomposition [7]. The scale space and pyramidal decomposition in particular are very well-known tools for image analysis and processing. Their focus is to exploit the high correlation existing in the neighboring pixels of an image and represent them in an efficient way. Observe, that this goal is very similar to the objective of sequential learning in which we want to characterize and learn the relationship between examples according to their labels. For the sake of clarity, we show the formulation in one dimension. However, it is a trivial task to extend it to an arbitrary dimensionality. Given the initial label sequence $y'^{(0)}$ of length L, and the concept of the floor integer rounding function defined as $\lfloor x \rfloor = \max\{n \in \mathcal{Z} \mid n \leq x\}$, each level of the decomposition is computed as follows,

$$y'^{(s+1)}_{\lfloor i/r \rfloor} = \sum_{m=-r_1}^{r_2} g^{(s)}(m) y'^{(s)}_{i+m},\ i \in [0, r, 2r, \ldots, \lfloor L/r^s \rfloor],\ r = r_1 + r_2 + 1,\ r_1, r_2 \geq 0 \quad (1)$$

where s defines the scale and $g^{(s)}(m)$ is a weighting function of length r defined around the origin of the window. Observe that with this definition the resulting label sequence is a reduced version of the label sequence at the previous scale where each new label is a weighted average of the former ones. From the point of view of machine learning, the weighing function $g^{(s)}(\cdot)$ can be interpreted as a prior model of the relationship between neighboring examples at one scale. This step is parameterized as $\theta = \{S, r_1, r_2, g^{(s)}(\cdot)\}$ in $J(y',\rho,\theta)$. Figure 2(a) shows the pyramidal decomposition with the following parameterization: the number of scales is $S = 3$, the relationship model used $g^s(\cdot)$ is a uniform distribution – no direction is preferred – with total length $r = 2$. The bottom sequence shows

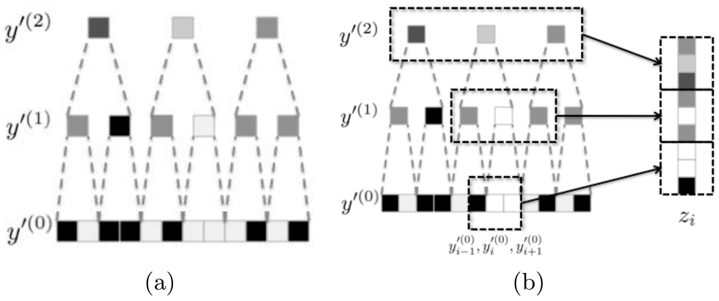


Fig. 2. Example of the process for obtaining z, (a) Multi-scale pyramid decomposition, (b) Feature creation by means of a scale-space window

the predicted labels $y'^{(0)} = y'$ for a binary classification problem (black/white). The middle step takes r predicted labels from $y'^{(0)}$ and applies the weighing function to obtain the values in $y'^{(1)}$. The same process is applied at each scale according to Eq. 1.

On the other hand, to define the support lattice, we use a scale-space sliding window. This is a window defined by w_1 and w_2 as left and right lengths, with a total size $w = w_1 + w_2 + 1$, defined with origin in the current example prediction in all the scales considered in the pyramidal decomposition – $\rho = \{w_1, w_2\}$. Thus, the output of $J(y', \rho, \theta)$ for S scales and a prior model window of length r and the support lattice parameterized by $\{w_1, w_2\}$ is given by,

$$z_i = J(y', \rho, \theta) =$$
$$\left(y'^{(0)}_{i-w_1}, .., y'^{(0)}_{i+w_2}, y'^{(1)}_{\lfloor i/r \rfloor - w_1}, .., y'^{(1)}_{\lfloor i/r \rfloor + w_2}, \ldots, y'^{(S)}_{\lfloor i/r^{S-1} \rfloor - w_1}, .., y'^{(S)}_{\lfloor i/2^{S-1} \rfloor + w_2}\right)$$

Figure 2(b) shows the second half of the process for a window of length $w = 3$ centered at the predicted label. At the bottom scale, we take the current predicted label with their neighboring ones $(y'^{(0)}_{i-1}, y'^{(0)}_{i}, y'^{(0)}_{i+1})$. Since higher scales are reduced versions of the original sequence, we have to find the correct indexing at each scale for obtaining its neighbors. This is done by casting the current index value using the following relationship: at scale s the index corresponding to i in the original sequence is given by $\lfloor i/r^s \rfloor$. Thus, for the sequence in the middle of Figure 2(b), we retrieve $\left(y'^{(1)}_{\lfloor i/2 \rfloor - 1}, y'^{(1)}_{\lfloor i/2 \rfloor}, y'^{(1)}_{\lfloor i/2 \rfloor + 1}\right)$. As a result, the final value of $J(y'_i, \rho, \theta)$ is

$$z_i = \left(y'^{(0)}_{i-1}, y'^{(0)}_{i}, y'^{(0)}_{i+1}, y'^{(1)}_{\lfloor i/2 \rfloor - 1}, y'^{(1)}_{\lfloor i/2 \rfloor}, y'^{(1)}_{\lfloor i/2 \rfloor + 1}, y'^{(2)}_{\lfloor i/2^2 \rfloor - 1}, y'^{(2)}_{\lfloor i/2^2 \rfloor}, y'^{(2)}_{\lfloor i/2^2 \rfloor + 1}\right)$$

Observe that the higher the number of scales is, the more predicted examples are considered. This feature allows to consider long distance interaction with a very small set of features while keeping a good short distance resolution. Figure 3 plots the number of features needed for observing a certain number of predicted labels (coverage). Different curves show the effect of altering the size of the

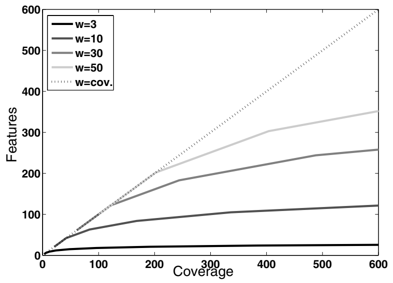


Fig. 3. Number of features needed for covering a certain number of predicted values for different window sizes of the support lattice

support window w. This parameter governs the trade-off between resolution and coverage. Thus, if the support window has a size of $w = 3$, we need 9 scales and a total of 27 features to capture information from 600 labels. However, details in long distance label interactions are lost. As w increases, the number of features increase also, but more complex and detailed label patterns can be captured. On the contrary, maximum interaction details are observed at the cost of using the same number of features as the coverage value. This trade-off allows the practitioner to consider different strategies according to the degree of correlation expected in the sequence. Moreover, if the neighboring interactions can be modeled, they can be used in this strategy by means of the prior $g^{(s)}(\cdot)$.

3 Experiments and Results

We test the proposed multi-scale sequential learning algorithm on two different problems according to the dimensionality of the relationship support lattice. The first is a FAQ categorization task, where each line from a FAQ text document has to be classified with labels like “header”, “question”,“answer” and “trailer”. The second experiment is a horse image classification task, i.e. the goal is to identify which of the image pixels belong to a horse taking into account their neighboring predictions.

3.1 Categorization of FAQ Documents

The FAQ categorization task has been frequently used in literature for benchmarking sequential learners [2] [8]. In this data set, three different computer science FAQ groups pages are used (ai-neural-nets, ai-general, aix). Each FAQ group consists of 5 to 7 long sequences of lines; each sequence corresponding to a single FAQ document. Each line is characterized using McCallun et al features [3], with 24 attributes that describe line characteristics with the respective class label. In total, each FAQ group contains between 8965 and 12757 labelled lines. This data set is multi-class, with 4 possible classes; in our experiments, for each of the three

Table 1. Error results, number of features and coverage values for different configurations of MSSL in the FAQ data sets

(r, S, w)	BASE	$(-, 1, 3)$	$(-, 1, 6)$	$(2, 2, 1)$	$(2, 3, 1)$	$(2, 4, 1)$	$(4, 4, 1)$
Features	-	7	13	6	9	12	12
Coverage	-	7	13	12	24	48	192
neural-netsA	7.0675	5.9855	5.8103	6.1495	5.624	4.9195	4.5257
neural-netsT	1.8067	1.4826	0.7825	1.3146	0.7159	0.5257	0.5199
Ai-GeneralA	8.2764	9.2944	10.3183	9.2636	9.1998	9.0374	10.3834
Ai-GeneralT	1.8916	1.6275	0.9392	1.7031	1.2716	0.6582	1.1964
Ai-AixA	9.7971	9.3519	9.9028	9.3307	9.3412	9.1311	9.5689
Ai-AixT	1.2553	0.8966	0.7493	0.9233	0.4754	0.2888	0.2662

groups we split the multi-class problem into two binary problems considering the following labels "answer" vs "not answer", and "tail" vs "not tail". The base classifier used in all the experiments is an AdaBoost with decision stumps reaching a maximum of 100 iterations. We performed a leave-one-out cross-validation for each sequence in each FAQ group – one sequence is used as testing and the rest of sequences are joined into one training sequence. In the first step of the sequential learning schema, a 5-fold crossvalidation using only the training data is performed to obtain the extended data set. Then, the first and second classifiers are trained with the whole training set and the extended training set respectively. Different configurations according to the (r, S, w) parameterization are compared.

Table 1 shows the results obtained for the FAQ experiments. Observe that the sequential learning approaches generally improve the accuracy performance except for the AI-GeneralA data set. Moreover, as the number of features increase the accuracy improves. Comparing SSL with the MSSL approach, it is worth noting that using a similar number of features, the multiscale counterparts achieve better accuracies. This seems to suggest that the data sets include some structure information that can only be captured at the largest scales. This idea is reinforced by the fact that for some of the test sets, the larger the coverage, the lower the error. With respect to the MSSL approaches, the more scales are used, the better the accuracy performance is. Finally, when the prior lattice size r is doubled – thus quadrupling the coverage by sacrificing details in the data sequence – we can see that on half of the data sets the accuracy improves.

3.2 Horse Image Classification

For the horse classification task, we used the Weizmann horse database [4], which consist of 328 images in gray scale. Each image is labelled according to the horse silhouette. In order to show the behavior of the different configurations independently of the feature set selected, we designed the experiment of classifying *dark* horses. To that end, we selected 8 images of clearly distinguishable dark horses and a test set of 29 images. As a pre-processing step, we rescale all the gray images to the same resolution 150×100. The feature vector is composed of just one value per pixel corresponding to the gray level intensity. All configurations use AdaBoost using decision stumps as base classifier with a maximum of 50 iterations.

For each image in the training set we perform a stratified sampling of 600 pixels per image. This data is used for training the first classifier using leave-one-image-out to produce the extended data set. Observe that in order to compute the neighbors of each pixel the whole left-out image is classified. Finally, both classifiers are trained using the same feature samples without and with the extended set respectively. This process is performed with different configurations using exactly the cross-validation samples. In order to avoid a biased result due to the random sampling, the whole experiment is repeated 10 times with different samplings. All experiments use a prior lattice of size $r = 2 \times 2$ with uniform distribution. Figure 4 illustrates the effect of different (S, w) configurations. The first column shows the input images, the second column displays the results of applying Adaboost, the third column shows the effect using the standard SSL which corresponds to a parameterization $(1, 5)$ of the generalized SSL and, finally, the last column shows the results using a $(6, 2)$ multi-scale parameterization. Observe that both configurations are comparable in terms of the number of features

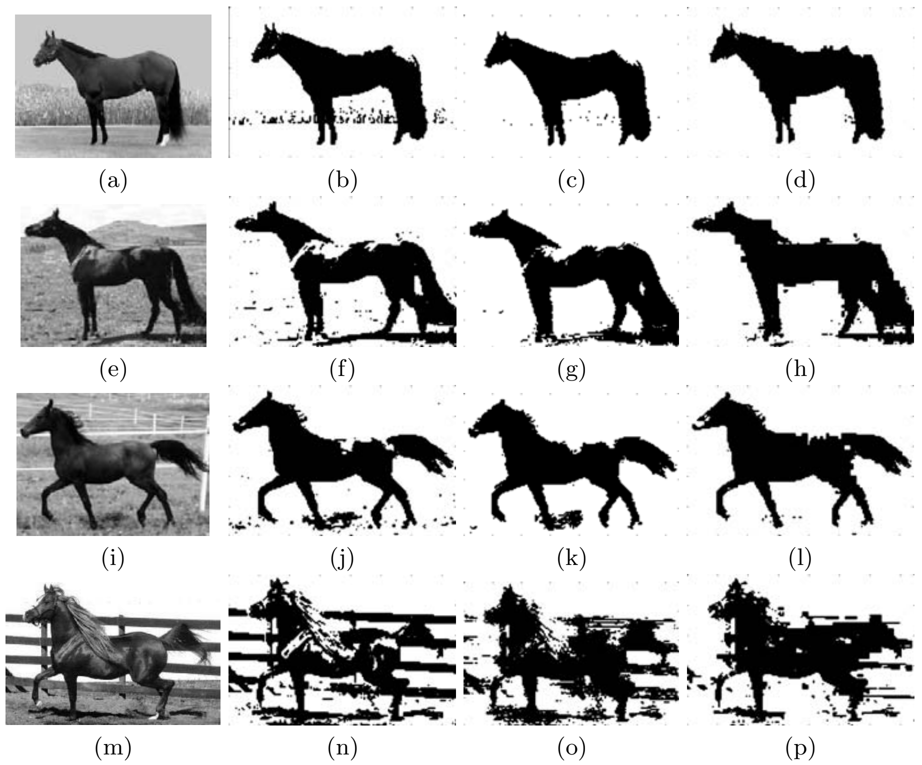


Fig. 4. Examples of the classification of dark horses using different strategies. First column: original gray level image; second column: results of Adaboost; third column: SSL with $w = 11 \times 11$ and; last column: MSSL using a $(6, 2)$ configuration.

Table 2. Improvement values, number of features and coverage value for different configurations of MSSL (including SSL) on the Weizmann data sets

(S, w)	$(1,3)$	$(1,5)$	$(1,7)$	$(3,1)$	$(6,1)$	$(6,2)$
Features	49	121	225	27	84	150
Coverage	49	121	225	36	288	800
Improv.	1.0789%	1.3091%	1.4765%	1.0975%	2.2499%	2.6538%

used. Figures 4(a)(b)(c)(d) show an example where the grass of the background creates some false positive spurious responses. As expected, both SSL and MSSL are capable of removing these artifacts. Figures 4(e)(f)(g)(h) show an example of label completion. The original classifier "beheads" the horse. If we observe the results in Figures 4(g)(h), standard SSL focuses on the details and barely fails to join head and body, and misses practically the whole back. However, the MSSL clearly completes the basic shape of the horse at the cost of losing some details. Figures 4(i)(j)(k)(l) show an example in which both former effects are present. Again observe that MSSL is able to reconstruct the back and tail of the horse and remove its shadow and the spurious responses more effectively. The last Figures 4(m)(n)(o)(p) are a clear example of context learning. The base classifier produces misleading responses associated to the background fence. With the MSSL we expect to learn the pixel relationships intrinsic to the true shape of the object. Observe in Figure 4(p) that most of the fence is removed successfully while preserving the basic shape of the horse and completing the missing labels in its neck and back.

Table 2 shows the figures result of the horse dyadic task. The last row of the table displays the average improvement of each algorithm with respect to the AdaBoost classifier. It is worth noting that the multiscale approach, using the parameters $(3, 1)$ performs similarly to the windowing approach with parameters $(1, 3)$ but using just 27 new features instead of 49. Moreover, the multiscale approach, using parameters $(6, 2)$, is almost two times better than the best of the standard SSL configurations using much less features. Finally, though the improvement slowly increases with the size of the window in the SSL approach, in the case of the MSSL approach, going from scale $S = 3$ to scale $S = 6$ with the same value $r = 1$ almost doubles the performance. This means that the increase in coverage helps the second classifier in capturing and exploiting long range interactions.

4 Conclusions

In this paper we generalize the stacked sequential learning process and stress the key role of the neighboring modeling. Additionally, we propose the use of a multi-scale approach, namely MSSL, that includes the classical SSL as a particular case and allows to capture long distance label interactions very efficiently. The method allows to have an explicit trade-off between the resolution of the interactions we desire to model and the number of features; very detailed interactions can be modeled in short distance or long distance patterns can be captured with a coarser resolution using the same number of features in the extended training set. We show that strategies mixing detailed interactions and long ranges can be defined at the cost of adding more features.

We show the behavior of the MSSL on two tasks. Results show that the MSSL approach qualitatively and quantitatively improves the SSL method considerably. Moreover, MSSL is extremely efficient in terms of the number of features used in the extended training set. In the authors' opinion MSSL is new technique that opens new lines of research in sequential learning allowing for the analysis of long distance interactions in a very efficient way.

Acknowledgments

This work has been supported in part by projects TIN2006-15308-C02, FIS PI061290, and CONSOLIDER-INGENIO CSD 2007-00018.

References

1. Dietterich, T.G.: Machine Learning for Sequential Data: A Review. In: Caelli, T.M., Amin, A., Duin, R.P.W., Kamel, M.S., de Ridder, D. (eds.) SPR 2002 and SSPR 2002. LNCS, vol. 2396, pp. 15–30. Springer, Heidelberg (2002)
2. Cohen, W.W., de Carvalho, V.R.: Stacked sequential learning. In: Proc. of IJCAI 2005, pp. 671–676 (2005)
3. McCallum, A., Freitag, D., Pereira, F.: Maximum entropy markov models for information extraction and segmentation. In: Proc. of ICML 2000, pp. 591–598 (2000)
4. Borenstein, E., Ullman, S.: Learning to segment. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3023, pp. 315–328. Springer, Heidelberg (2004)
5. Wolpert, D.H.: Stacked generalization. Neural Networks 5(2), 241–259 (1992)
6. Lafferty, J.D., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. of ICML 2001, pp. 282–289 (2001)
7. Burt, P., Adelson, E.: The laplacian pyramid as a compact image code. IEEE Transactions on Communications 31(4), 532–540 (1983)
8. Dietterich, T.G., Ashenfelter, A., Bulatov, Y.: Training conditional random elds via gradient tree boosting. In: Proc. of the 21th ICML (2004)
9. Bottou, L., Bengio, Y., LeCun, Y.: Global training of document processing systems using graph transformer networks. In: CVPR, pp. 489–494. IEEE Computer Society, Los Alamitos (1997)

Unsupervised Hierarchical Weighted Multi-segmenter

Michal Haindl[1,2], Stanislav Mikeš[1], and Pavel Pudil[1,2]

[1] Institute of Information Theory and Automation
Academy of Sciences CR, Prague, Czech Republic
[2] Faculty of Management, University of Economics
Jindřichův Hradec, Czech Republic
{haindl,xaos}@utia.cz

Abstract. An unsupervised multi-spectral, multi-resolution, multiple-segmenter for textured images with unknown number of classes is presented. The segmenter is based on a weighted combination of several unsupervised segmentation results, each in different resolution, using the modified sum rule. Multi-spectral textured image mosaics are locally represented by four causal directional multi-spectral random field models recursively evaluated for each pixel. The single-resolution segmentation part of the algorithm is based on the underlying Gaussian mixture model and starts with an over segmented initial estimation which is adaptively modified until the optimal number of homogeneous texture segments is reached. The performance of the presented method is extensively tested on the Prague segmentation benchmark using the commonest segmentation criteria and compares favourably with several leading alternative image segmentation methods.

1 Introduction

Segmentation is the fundamental process which partitions a data space into meaningful salient regions. Image segmentation essentially affects the overall performance of any automated image analysis system thus its quality is of the utmost importance. Image regions, homogeneous with respect to some usually textural or colour measure, which result from a segmentation algorithm are analysed in subsequent interpretation steps. Texture-based image segmentation is area of intense research activity in recent years and many algorithms were published in consequence of all this effort. These methods are usually categorised [1] as region-based, boundary-based, or as a hybrid of the two. Different published methods are difficult to compare because of lack of a comprehensive analysis together with accessible experimental data, however available results indicate that the ill-defined texture segmentation problem is still far from being satisfactorily solved. Spatial interaction models and especially Markov random fields-based models are increasingly popular for texture representation [1,2], etc. Several researchers dealt with the difficult problem of unsupervised segmentation using these models see for example [3,4,5] or [6,7,8].

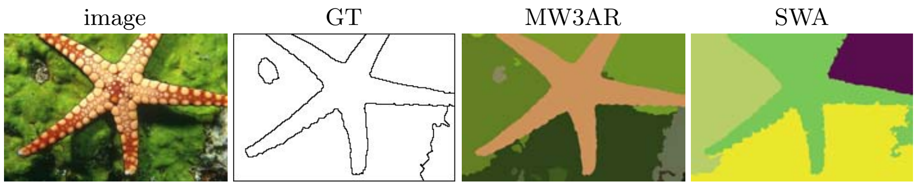


Fig. 1. Selected Berkeley benchmark image, ground truth from the benchmark and the segmentation results from the presented method (MW3AR) and SWA [9]

The concept of decision fusion [10] for high-performance pattern recognition is well known and widely accepted in the area of supervised classification where (often very diverse) classification technologies, each providing complementary sources of information about class membership, can be integrated to provide more accurate, robust and reliable classification decisions than the single classifier applications.

Similar advantages can be expected and achieved [8] also for the unsupervised segmentation applications. However, a direct unsupervised application of the supervised classifiers fusion idea is complicated with unknown number of data hidden classes and consequently a different number of segmented regions in segmentation results to be fused. This paper exploits above advantages by combining several unsupervised segmenters of the same type but with different feature sets.

2 Combination of Multiple Segmenters

The proposed method (MW3AR) combines segmentation results from different resolution. We assume to down-sample input image Y into M different resolutions $Y^{(m)} = \downarrow^{\iota_m} Y$ with sampling factors ι_m $m = 1, \ldots, M$ identical in both horizontal and vertical directions and $Y^{(1)} = Y$. Local texture for each pixel $Y_r^{(m)}$ is represented the 3D simultaneous causal autoregressive random field model (CAR) parameter space Θ_r (4) and modelled by the Gaussian mixture model (5),(6).

2.1 Single-Resolution Texture Model

Static smooth multi-spectral textures require three dimensional models for adequate representation. We assume that single multi-spectral textures can be locally modelled using a 3D simultaneous causal autoregressive random field model (CAR). This model can be expressed as a stationary causal uncorrelated noise driven 3D autoregressive process [11]:

$$Y_r = \gamma X_r + e_r \quad , \tag{1}$$

where $\gamma = [A_1, \ldots, A_\eta]$ is the $d \times d\eta$ parameter matrix, d is the number of spectral bands, I_r^c is a causal neighborhood index set with $\eta = card(I_r^c)$ and

e_r is a white Gaussian noise vector with zero mean and a constant but unknown covariance, X_r is a corresponding vector of the contextual neighbours Y_{r-s} and $r, r-1, \ldots$ is a chosen direction of movement on the image index lattice I. The selection of an appropriate CAR model support (I_r^c) is important to obtain good texture representation but less important for segmentation. The optimal neighbourhood as well as the Bayesian parameters estimation of a CAR model can be found analytically under few additional and acceptable assumptions using the Bayesian approach (see details in [11]). The recursive Bayesian parameter estimation of the CAR model is [11]:

$$\hat{\gamma}_{r-1}^T = \hat{\gamma}_{r-2}^T + \frac{V_{x(r-2)}^{-1} X_{r-1} (Y_{r-1} - \hat{\gamma}_{r-2} X_{r-1})^T}{(1 + X_{r-1}^T V_{x(r-2)}^{-1} X_{r-1})} \quad , \tag{2}$$

where $V_{x(r-1)} = \sum_{k=1}^{r-1} X_k X_k^T + V_{x(0)}$. Local texture for each pixel is represented by four parametric vectors. Each vector contains local estimations of the CAR model parameters. These models have identical contextual neighbourhood I_r^c but they differ in their major movement direction (top-down, bottom-up, rightward, leftward), i.e.,

$$\tilde{\gamma}_r^T = \{\hat{\gamma}_r^t, \hat{\gamma}_r^b, \hat{\gamma}_r^r, \hat{\gamma}_r^l\}^T \quad . \tag{3}$$

The parametric space $\tilde{\gamma}$ is subsequently smooth out, rearranged into a vector and its dimensionality is reduced using the Karhunen-Loeve feature extraction $(\bar{\gamma})$. Finally we add the average local spectral values ζ_r to the resulting feature vector (Θ_r).

2.2 Mixture Based Segmentation

Multi-spectral texture segmentation is done by clustering in the CAR parameter space Θ defined on the lattice I where

$$\Theta_r = [\bar{\gamma}_r, \zeta_r]^T \tag{4}$$

is the modified local parameter vector (3) computed for the lattice location r. We assume that this parametric space can be represented using the Gaussian mixture model (GM) with diagonal covariance matrices due to the previous CAR parametric space decorrelation. The Gaussian mixture model for CAR parametric representation at the m-th resolution $(m = 1, \ldots, M)$ is as follows:

$$p(\Theta_r^{(m)}) = \sum_{i=1}^{K^{(m)}} p_i^{(m)} \, p(\Theta_r^{(m)} \,|\, \nu_i^{(m)}, \Sigma_i^{(m)}) \quad , \tag{5}$$

$$p(\Theta_r^{(m)} \,|\, \nu_i^{(m)}, \Sigma_i^{(m)}) = \frac{|\Sigma_i^{(m)}|^{-\frac{1}{2}}}{(2\pi)^{\frac{d}{2}}} \, e^{-\frac{(\Theta_r^{(m)} - \nu_i^{(m)})^T (\Sigma_i^{(m)})^{-1} (\Theta_r^{(m)} - \nu_i^{(m)})}{2}} \quad . \tag{6}$$

The mixture model equations (5),(6) are solved using a modified EM algorithm.

Initialization. The algorithm is initialised using $\nu_i^{(m)}, \Sigma_i^{(m)}$ statistics for each resolution m estimated from the corresponding thematic maps in two subsequent steps:

1. refining direction
   $\nu_i^{(m-1)} \left(\forall \Theta_r^{(m-1)} : \, r \in \uparrow \Xi_i^{(m)}\right)$, $\Sigma_i^{(m-1)} \left(\forall \Theta_r^{(m-1)} : \, r \in \uparrow \Xi_i^{(m)}\right)$
   $m = M+1, M, \ldots, 2$ $i = 1, \ldots, K^{(m)}$,
2. coarsening direction
   $\nu_i^{(m)} \left(\forall \Theta_r^{(m)} : \, r \in \downarrow \Xi_i^{(m-1)}\right)$, $\Sigma_i^{(m)} \left(\forall \Theta_r^{(m)} : \, r \in \downarrow \Xi_i^{(m-1)}\right)$
   $m = 2, 3, \ldots, M$ $i = 1, \ldots, K^{(m)}$,

where $\Xi_i^{(m)} \subset I \; \forall m, i$, and the first initialisation thematic map $\Xi_i^{(M+1)}$ is approximated by the rectangular subimages obtained by regular division of the input texture mosaic. All the subsequent refining step are initialised from the preceding coarser resolution upsampled thematic maps. The final initialisation results from the second coarsening direction where the gradually coarsening segmentations are initialised using the preceding downsampled thematic maps. For each possible couple of components the Kullback Leibler divergence

$$D\left(p(\Theta_r \,|\, \nu_i, \Sigma_i) \,||\, p(\Theta_r \,|\, \nu_j, \Sigma_j)\right) = \int_{\Omega} p(\Theta_r \,|\, \nu_i, \Sigma_i) \, \log \left(\frac{p(\Theta_r \,|\, \nu_i, \Sigma_i)}{p(\Theta_r \,|\, \nu_j, \Sigma_j)} \right) d\Theta_r$$

is evaluated and the most similar components, i.e.,

$$\{i, j\} = \arg\min_{k,l} D\left(p(\Theta_r \,|\, \nu_l, \Sigma_l) \,||\, p(\Theta_r \,|\, \nu_k, \Sigma_k)\right)$$

are merged together in each initialisation step. This initialisation results in K_{ini} subimages and recomputed statistics ν_i, Σ_i . $K_{ini} > K$ where K is the optimal number of textured segments to be found by the algorithm.

Two steps of the EM algorithm are repeating after initialisation. The components with smaller weights than a fixed threshold $(p_j < \frac{0.1}{K_{ini}})$ are eliminated. For every pair of components we estimate their Kullback Leibler divergence. From the most similar couple, the component with the weight smaller than the threshold is merged to its stronger partner and all statistics are actualised using the EM algorithm. The algorithm stops when either the likelihood function has negligible increase $(\mathcal{L}_t - \mathcal{L}_{t-1} < 0.05)$ or the maximum iteration number threshold is reached.

2.3 Resulting Mixture Probabilities

Resulting mixture model probabilities are mapped to the original fine resolution image space for all $m = 1, \ldots, M$ mixture submodels ((5),(6)). The M cooperating segmenters deliver their class response in the form of conditional probabilities. Each segmenter produces a preference list based on the mixture component probabilities of a particular pixel belonging a particular class, together with a set of confidence measurement values generated in the original decision-making process.

Single-segmenters Correspondence. Single-resolution segmentation results cannot be combined without knowledge of the mutual correspondence between regions in all different-resolution segmentation probabilistic mixture component maps ($K^1 \times \sum_{m=2}^{M} K^m$ combinations). Mutual assignments of two probabilistic maps are solved by using the Munkre's assignment algorithm [8] which finds the minimal cost assignment

$$g : A \mapsto B, \; \sum_{\alpha \in A} f(\alpha, g(\alpha))$$

between sets A, B, $|A| = |B| = n$ given the cost function $f(\alpha, \beta)$, $\alpha \in A, \beta \in B$. α corresponds to the fine resolution probabilistic maps, β corresponds to downsampled probabilistic maps and $f(\alpha, \beta)$ is the Kullback Leibler divergence between probabilistic maps. The algorithm has polynomial complexity instead of exponential for the exhaustive search.

Final Parametric Space. The parametric vectors representing texture mosaic pixels are assigned to the clusters based on our modification of the sum rule according to the highest component probabilities, i.e., Y_r is assigned to the cluster ω_{j^*} if

$$\pi_{r,j^*} = max_j \sum_{s \in I_r} w_s \left(\sum_{m=1}^{M} \frac{p^2(\Theta_{r-s}^{(m)} \,|\, \nu_j^{(m)}, \Sigma_j^{(m)})}{\sum_{i=1}^{M} p(\Theta_{r-s}^{(i)} \,|\, \nu_j^{(i)}, \Sigma_j^{(i)})} \right) ,$$

where w_s are fixed distance-based weights, I_r is a rectangular neighbourhood and $\pi_{r,j^*} > \pi_{thre}$ (otherwise the pixel is unclassified). The area of single cluster blobs is evaluated in the post-processing thematic map filtration step. Regions with similar statistics are merged. Thematic map blobs with area smaller than a given threshold are attached to its neighbour with the highest similarity value.

3 Experimental Results

The algorithm was tested on natural colour textures mosaics from the Prague Texture Segmentation Data-Generator and Benchmark (http://mosaic.utia.cas.cz) [12]. The benchmark test mosaics layouts and each cell texture membership are randomly generated and filled with colour textures from the large (more than 1000 high resolution colour textures) Prague colour texture database. The benchmark ranks segmentation algorithms according to a chosen criterion. There are implemented twenty seven most frequented evaluation criteria categorised into four criteria groups – region-based [12], pixel-wise [12], clustering comparison criteria, and consistency measures [12]. The region-based [12] performance criteria mutually compare ground truth (GT) image regions with the corresponding machine segmented regions (MS). The pixel-wise criteria group contains the most frequented classification criteria such as the omission and commission errors, class accuracy, recall, precision, etc. Finally the last two criteria sets incorporate the global and local consistency errors [12] and three clustering comparison criteria.

Table 1. Benchmark criteria: CS = correct segmentation; OS = over-segmentation; US = under-segmentation; ME = missed error; NE = noise error; O = omission error; C = commission error; CA = class accuracy; CO = recall - correct assignment; CC = precision - object accuracy; I. = type I error; II. = type II error; EA = mean class accuracy estimate; MS = mapping score; RM = root mean square proportion estimation error; CI = comparison index; GCE = Global Consistency Error; LCE = Local Consistency Error; dM = Mirkin metric; dD = Van Dongen metric; dVI = variation of information;

	Benchmark – Colour											
	MW3AR	TFR / KLD [13]	TFR [14]	AR3D + EM multi [8]	AR3D + EM [7]	GMRF + EM [6]	HGSE [15]	EG-BIS [16]	JSEG [17]	SWA def_pa [9]	Blob-world [18]	EDI-SON [19]
CS	53.04	51.25	46.13	43.22	37.42	31.93	29.81	28.78	27.47	27.06	21.01	12.68
OS	59.53	5.84	2.37	49.27	59.53	53.27	10.69	19.69	38.62	50.21	7.33	86.91
US	3.20	7.16	23.99	16.55	8.86	11.24	33.76	39.15	5.04	4.53	9.30	0.00
ME	5.63	31.64	26.70	10.30	12.54	14.97	26.89	20.42	35.00	25.76	59.55	2.48
NE	6.96	31.38	25.23	12.56	13.14	16.91	25.04	21.54	35.50	27.50	61.68	4.68
O	19.32	19.65	28.73	21.99	34.32	33.61	48.94	44.35	37.94	33.01	41.45	73.17
C	86.19	9.67	12.50	87.38	100.00	100.00	32.39	82.87	92.77	85.19	58.94	100.00
CA	71.89	67.45	61.32	64.51	59.46	57.91	49.60	51.10	55.29	54.84	46.23	31.19
CO	74.66	76.40	73.00	71.00	64.81	63.51	63.37	64.12	61.81	60.67	56.04	31.55
CC	95.04	81.12	68.91	90.14	91.79	89.26	66.09	72.73	87.70	88.17	73.62	98.09
I.	25.34	23.60	27.00	29.00	35.19	36.49	36.63	35.88	38.19	39.33	43.96	68.45
II.	0.74	4.09	8.56	3.79	3.39	3.14	13.51	7.59	3.66	2.11	6.72	0.24
EA	80.43	75.80	68.62	73.90	69.60	68.41	58.74	59.88	66.74	66.94	58.37	41.29
MS	71.78	65.19	59.76	64.47	58.89	57.42	46.63	49.03	55.14	53.71	40.36	31.13
RM	3.09	7.21	8.61	4.55	4.88	4.86	13.31	8.38	4.96	6.11	7.96	3.21
CI	82.43	77.21	69.73	76.51	73.15	71.80	61.17	63.11	70.27	70.32	61.31	50.29
GCE	8.17	20.35	15.52	15.31	12.13	16.03	16.75	16.64	18.45	17.27	31.16	3.55
LCE	5.78	14.36	12.03	7.97	6.69	7.31	10.46	8.97	11.64	11.49	23.19	3.44
dM	8.97	12.64	17.47	13.51	15.43	15.27	27.95	19.72	15.19	13.68	20.03	16.84
dD	14.78	18.01	18.21	16.87	19.76	20.63	22.90	21.29	23.38	24.20	31.11	35.37
dVI	16.67	14.06	13.04	16.11	17.10	17.32	12.83	13.79	17.37	17.16	15.84	25.65

Tab.1 compares the overall benchmark performance of the proposed algorithm MW3AR ($M = 5, \iota_1 = 1, \iota_2 = 1.5, \iota_3 = 2$, with the Blobworld [18], JSEG [17], Edison [19], TFR/KLD [14], SWA [9], EGBIS [16], HGS [15], and our previously published methods AR3D-multi [8], GMRF [6], AR3D [7], respectively. MW3AR demonstrates a significant improvement (e.g. 23 % for the correct segmentation CS criterion) over our previously published unsupervised multi-segmenter AR3D-multi [8].

These results illustrated in Figs.1,2,3 and Tab.1 demonstrate very good pixel-wise, correct region segmentation, missed error, noise error, and undersegmentation properties of our method while the oversegmentation results are slightly

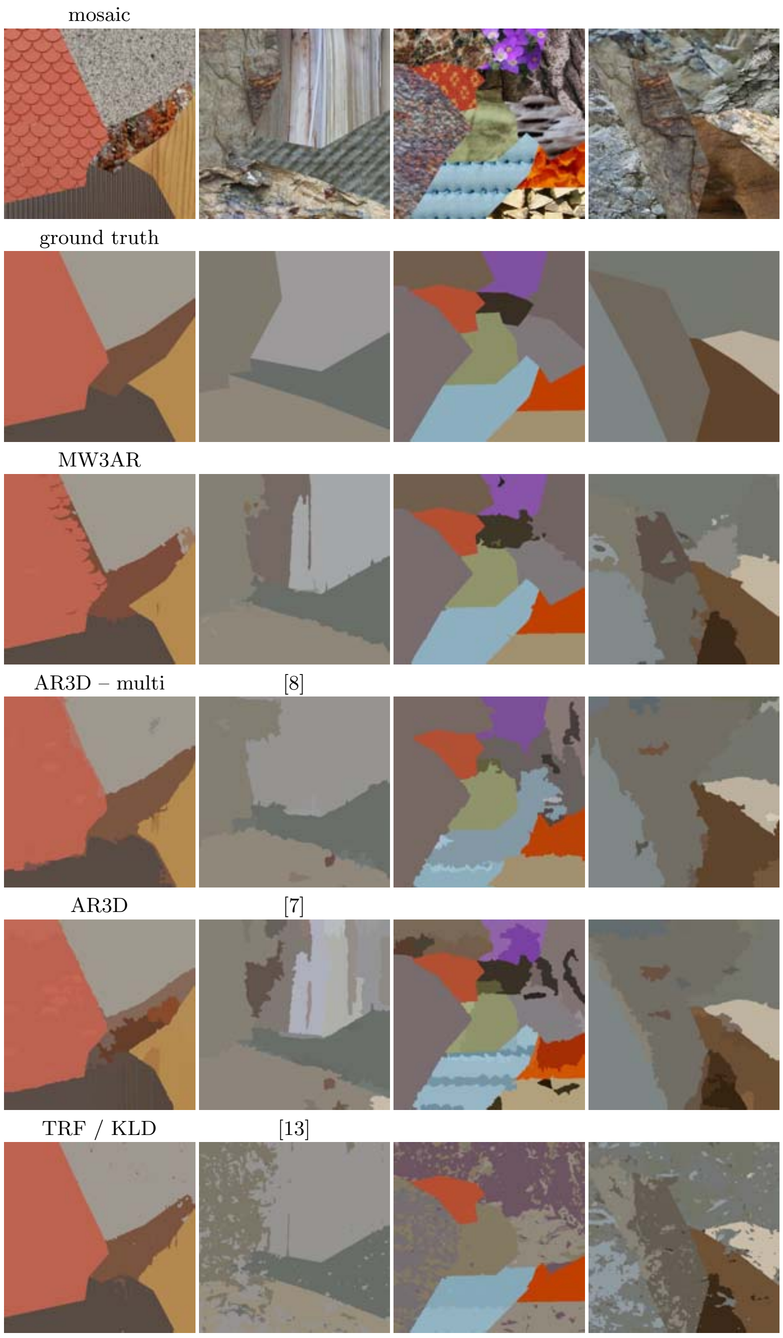


Fig. 2. Selected experimental texture mosaics, ground truth from the benchmark and the corresponding segmentation results

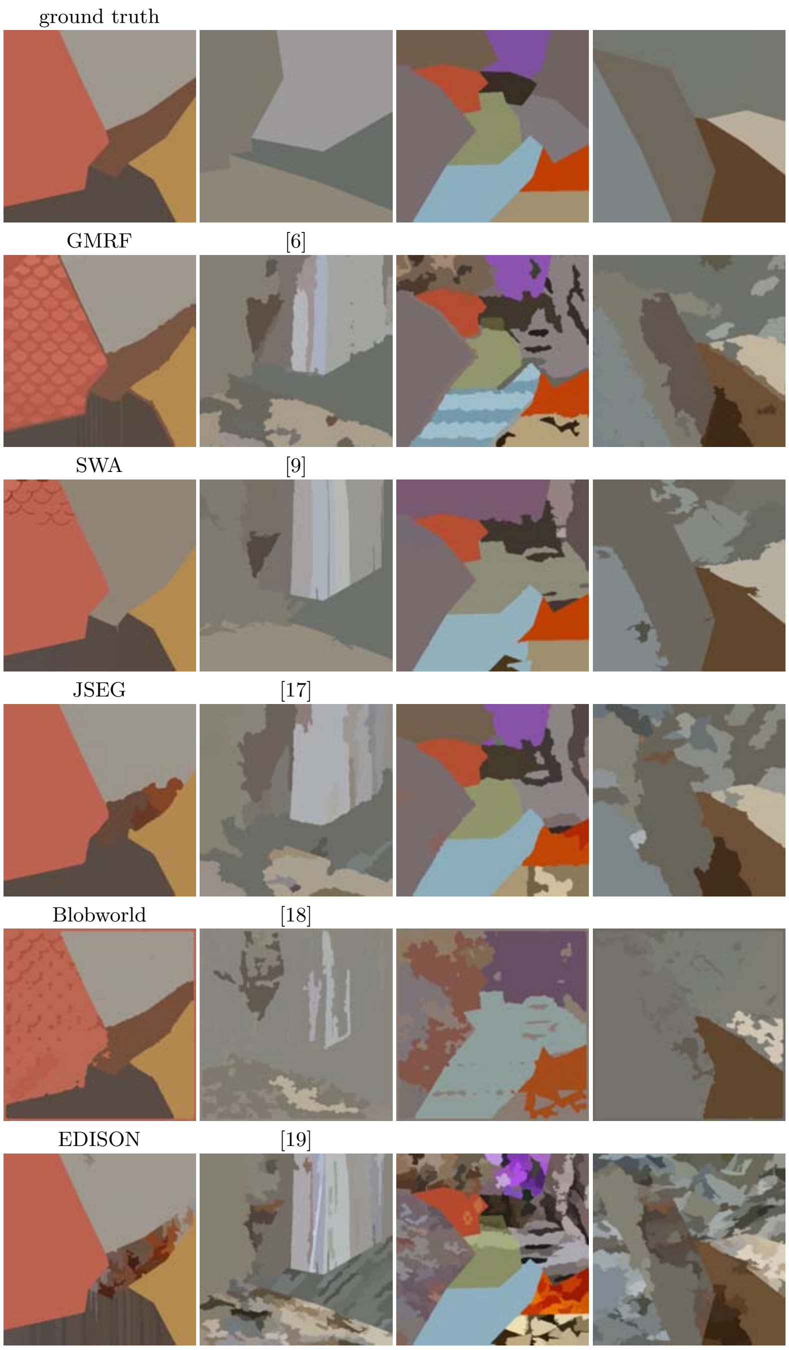


Fig. 3. Selected ground truth from the benchmark and the corresponding segmentation results

worse and dVI results are only average. For all the pixel-wise criteria or the consistency measures our method is among the best ones. The table demonstrates improvement of the presented multi-segmenter method over our previous multi-segmenter [8] and its single-segmenter version published earlier [7] in most benchmark criteria.

Figs.2,3 and show four selected 512×512 experimental benchmark mosaics created from four to eleven natural colour textures. The last four or five rows on these figures demonstrate comparative results from the eight alternative leading algorithms. Hard natural textures were chosen rather than synthesised (for example using Markov random field models) ones because they are expected to be more difficult for the underlying segmentation model. The third row on Fig.2 demonstrates robust behaviour of our CAR3D-multi algorithm but also infrequent algorithm failures producing the oversegmented thematic map for some textures. Such failures can be reduced by a more elaborate postprocessing step. The TFR/KLD [14], AR3D [7], GMRF [6], SWA [9], EGBIS [16], JSEG [17], Blobworld [18], HGS [15], and Edison [19], algorithms on these data performed mostly worse as can be seen in their corresponding rows on Figs.2,3 some areas are undersegmented while other parts of the mosaics are oversegmented. Fig.2 illustrates also the improvement of the multi-segmenter version of the algorithm at the cost of slight increase of computational complexity. These results can be further improved by sophisticated postprocessing and by the optimisation of the directional models contextual neighbourhoods.

4 Conclusions

We proposed a significant improvement of our previously published unsupervised multi-segmenter. The MW3AR segmenter is computationally efficient, noise resilient and robust method for unsupervised textured image segmentation with unknown number of classes based on the underlying CAR and GM texture models. The algorithm is very fast, despite of using the random field type data representation, due to its efficient recursive parameter estimation of the underlying models and therefore is much faster than the usual Markov chain Monte Carlo estimation approach required for these image representations. Usual drawback of most segmentation methods is their application dependent parameters to be experimentally estimated. Our method requires only a contextual neighbourhood selection and two additional thresholds. The method's performance is demonstrated on the extensive benchmark tests on natural texture mosaics. It performs favourably compared with eight alternative leading segmentation algorithms. Our method accomplishes very good segmentation results also on natural images from the Berkeley segmentation benchmark as well as on remote sensing images.

Acknowledgements

This research was supported by the grant GAČR 102/08/0593 and partially by the MŠMT grants 1M0572 DAR, 2C06019.

References

1. Reed, T.R., du Buf, J.M.H.: A review of recent texture segmentation and feature extraction techniques. CVGIP–Image Understanding 57(3), 359–372 (1993)
2. Haindl, M.: Texture synthesis. CWI Quarterly 4(4), 305–331 (1991)
3. Panjwani, D., Healey, G.: Markov random field models for unsupervised segmentation of textured color images. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(10), 939–954 (1995)
4. Manjunath, B., Chellapa, R.: Unsupervised texture segmentation using markov random field models. IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 478–482 (1991)
5. Haindl, M.: Texture segmentation using recursive markov random field parameter estimation. In: Proceedings of the 11th Scandinavian Conference on Image Analysis, Lyngby, Denmark, Pattern Recognition Society of Denmark, pp. 771–776 (1999)
6. Haindl, M., Mikeš, S.: Model-based texture segmentation. In: Campilho, A.C., Kamel, M.S. (eds.) ICIAR 2004. LNCS, vol. 3212, pp. 306–313. Springer, Heidelberg (2004)
7. Haindl, M., Mikeš, S.: Unsupervised texture segmentation using multispectral modelling approach. In: Proceedings of the 18th Int. Conf. on Pattern Recognition, ICPR 2006, vol. II, pp. 203–206. IEEE Computer Society, Los Alamitos (2006)
8. Haindl, M., Mikes, S.: Unsupervised texture segmentation using multiple segmenters strategy. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 210–219. Springer, Heidelberg (2007)
9. Sharon, E., Galun, M., Sharon, D., Basri, R., Brandt, A.: Hierarchy and adaptivity in segmenting visual scenes. Nature 442(7104), 719–846 (2006)
10. Kittler, J., Hojjatoleslami, A., Windeatt, T.: Weighting factors in multiple expert fusion. In: Proc. BMVC, BMVA, pp. 41–50 (1997)
11. Haindl, M., Šimberová, S.: A Multispectral Image Line Reconstruction Method. In: Theory & Applications of Image Analysis, pp. 306–315. World Scientific Publishing Co., Singapore (1992)
12. Haindl, M., Mikeš, S.: Texture segmentation benchmark. In: Lovell, B., Laurendeau, D., Duin, R. (eds.) Proceedings of the 19th Int. Conf. on Pattern Recognition, ICPR 2008. IEEE Computer Society, Los Alamitos (2008)
13. Scarpa, G., Haindl, M., Zerubia, J.: A hierarchical finite-state model for texture segmentation. In: ICASSP 2007. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, vol. I, pp. 1209–1212. IEEE, Los Alamitos (2007)
14. Scarpa, G., Haindl, M.: Unsupervised texture segmentation by spectral-spatial-independent clustering. In: Proc. of the 18th Int. Conf. on Pattern Recognition, ICPR 2006, vol. II, pp. 151–154. IEEE Computer Society, Los Alamitos (2006)
15. Hoang, M.A., Geusebroek, J.M., Smeulders, A.W.: Color texture measurement and segmentation. Signal Processing 85(2), 265–275 (2005)
16. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. IJCV 59(2), 167–181 (2004)

17. Deng, Y., Manjunath, B.: Unsupervised segmentation of color-texture regions in images and video. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(8), 800–810 (2001)
18. Carson, C., Thomas, M., Belongie, S., Hellerstein, J.M., Malik, J.: Blobworld: A system for region-based image indexing and retrieval. In: Third International Conference on Visual Information Systems. Springer, Heidelberg (1999)
19. Christoudias, C., Georgescu, B., Meer, P.: Synergism in low level vision. In: Proceedings of the 16th Int. Conf. on Pattern Recognition, vol. 4, pp. 150–155. IEEE Computer Society, Los Alamitos (2002)

Ant Clustering Using Ensembles of Partitions

Yuhua Gu, Lawrence O. Hall, and Dmitry B. Goldgof

Dept. of Computer Science & Engineering
University of South Florida
Tampa, FL 33620, U.S.A.
{hall,goldgof,yuhua}@cse.usf.edu
http://www.cse.usf.edu

Abstract. Classical clustering algorithms require a predefined number of cluster centers. They are often very sensitive to initialization, which can result in very different clustering results. We present a two-phase algorithm which is a combination of a new ant based algorithm and a nonnegative matrix factorization-based consensus clustering algorithm. Ant clustering approaches can and do find the number of clusters as well as the data partition. However, they are very sensitive to both initial conditions and select parameters. Here, we show that using an ensemble of ant partitions and NMF to combine them we can find both the "right" number of clusters and a good data partition. Experiments were done with ten data sets. We conducted a wide range of comparisons that demonstrate the effectiveness of this new approach.

Keywords: Clustering, Ant-based clustering, Fuzzy c-means, Consensus clustering, NMF, Ensembles.

1 Introduction

The main aim of clustering is to divide a data set into groups so that the data within a group are sufficiently similar while the data belonging to different groups are dissimilar. Two big drawbacks of many existing clustering algorithms such as the Fuzzy C means (FCM) algorithm, K-means algorithm and Expectation Maximization (EM) algorithm are that they require prior knowledge of the number of clusters for the data and are very sensitive to cluster center initialization, since the search for a partition is based on the hill climbing heuristic [1,2]. Many clustering algorithms [3,4,5,8,19] developed using the principles of Swarm Intelligence are considered distributed, flexible and robust. The algorithms simulate the co-operative and stochastic iterative behavior of a colony of ants, they are based on direct or indirect feedback with relatively simple agents [18].

The behavior of ant colonies has inspired the development of various clustering techniques: (1) Several species of ants cluster their corpses into cemeteries in order to clean up their nests. (2) Besides nest cleaning, another interesting approach is chemical recognition of nest mates. The ant algorithm in this paper is based on chemical odor. In [8,20], an ant clustering algorithm was proposed by correspondence of a nest to a cluster and a chemical odor to a cluster label according to some ant behavioral rules. In ant clustering the ants are themselves the data, which means the number of examples is exactly the number of ants. We found if one dataset is not well separated in feature space, there are always some data left unlabeled after the ant's refining phase. In order to finish clustering, the Fuzzy C means algorithm was used on only the unclassified examples. The number of clusters and initialization for FCM came directly from ant clustering. Lastly, we merged the centroids obtained from both stages and labeled the data. We call the first stage ANTFCM. Ant clustering is randomly self-organized, meaning each run may result in a different partition, which increases the difficulty of finding the optimal solution and evaluating the partition quality. The creation of different partitions can be treated as an ensemble of partitions. We used consensus clustering to merge the ensemble of data partition solutions here, which resulted in the ability to choose the right number of clusters and an overall partition that better modeled the true distribution of data. A number of approaches have recently been developed to solve consensus clustering problems [9,10,11,12,13,14,15]. We used the nonnegative matrix factorization (NMF) [12,13,15] algorithm as a consensus clustering algorithm as the 2nd stage of our algorithm. NMF is very simple to implement and often yields good results [16]. It has been found to outperform many other consensus methods [12].

2 Ant Clustering and NMF-Based Consensus Clustering

2.1 Ant Clustering

We first describe the ant clustering algorithm proposed in [8] by Labroche. Zaharie and Zamfirache [20] proposed ant based clustering for dealing with noise based on Labroche's method [8]. Our definitions in this paper mainly come from those of [20]. Let x_1,...,x_n be the set of data, where n is the number of ants:

– x_i is associated with an example, L_i is a label, which is a natural number used to identify a cluster. It is initially set to be 0 meaning that the data has not been assigned to a cluster,
– T_i is a similarity threshold. A_i is age, which counts the number of meetings in which the ant has participated,
– M_i defines an adaptive parameter, which measures the ant's perception of its nest's size, M_i^+ defines another adaptive parameter, which measures the ant's perception of its acceptance degree by the other members of its nest,
– $N_{cluster} = \frac{N_i}{n}$, where N_i is the number of examples in the i^{th} cluster ($N_{cluster}$ is the ratio of data belonging to the cluster).

Ant clustering has three main phases: the threshold learning phase, the random meeting phase and the refine cluster phase. The first phase is mainly to estimate the value of the similarity threshold T_i for each ant i. The similarities between an ant i and other randomly selected k_t ants are computed and the maximum $(max(S(i,.)))$ and the average $(< S(i,.) >)$ of these similarities are obtained. The estimation of T_i then is defined by $(maxS(i,.) + < S(i,.) >)/2$,

$S(i,j) = \frac{1}{s}\sum_{k=1}^{s}(1 - \frac{|x_i^k - x_j^k|}{max(x^k)-min(x^k)})$ where s is the number of features. $S(i,j)$ has proved to be a reasonable similarity function [8,20]). In the random meetings phase, random pairs, (i, j), of distinct ants are selected k_m times. The similarity $S(i,j)$ for each pair is computed and for each pair of ants it is determined whether the ants accept each other. The definition of an acceptance between ants i and j is Accept(i,j)=True if and only if $S(i,j) > T_i$ and $S(i,j) > T_j$, Accept(i,j)=False otherwise. In the cluster refining stage, the clusters with a small number of elements and low M_i^+ values are eliminated and they are merged with other clusters based on their similarities and on the values of M_i and M_i^+.

Behavioral rules 1-5 below [20] are applied to each meeting of ant pairs (i, j).

Rule 1: new nest creation. If Accept (i, j) is True and $L_i = L_j = 0$ then $L_i := L_{max} + 1, L_j := L_{max} + 1$ where L_{max} is the maximal value of the labels assigned up to the current step.
Rule 2: if Accept(i,j) is True and $L_i = 0, L_j! = 0$ then $L_i := L_j$, which means put an ant into an existing nest.
Rule 3: positive meeting between two nestmates. If Accept (i, j) is True and $L_i = L_j! = 0$ then increase M_i, M_i^+, M_j, M_j^+.
Rule 4: negative meeting between two nestmates. If Accept (i, j) is False and $L_i = L_j! = 0$ then the ant with the smaller acceptance degree is excluded from its nest, all parameters related to this ant are set to 0. The parameter M of the other ant is increased while the parameter M^+ is decreased.
Rule 5: meeting between ants belonging to different nests. If Accept (i, j) is True and $L_i! = L_j! = 0$ then the ant with lower M is included in the nest of the other ant and the corresponding M_i, M_j decreased.

The increasing and decreasing factors in Rules 4 and 5 are based on the following relations:

$$increase(\nu) = (1-\alpha)\nu + \alpha, decrease(\nu) = (1-\alpha)\nu \tag{1}$$

where $\alpha \in (0,1)$.

2.2 ANTFCM and NMF-Based Consensus Clustering

The general structure of the ant clustering algorithm [20] and how to combine it with Fuzzy C means clustering is shown in Fig. 1.

To discuss NMF-based consensus clustering, we first must introduce some notation [12,15]. Let $X = \{x_1, x_2, ..., x_n\}$ be a set of n examples, suppose we have a set of T clusterings (or partitions) $P = \{P^1, P^2, ..., P^T\}$, each partition $P^t, (t = 1, ..., T)$ includes a set of clusters $C^t = \{C_1^t, C_2^t, ..., C_k^t\}$, where k represents the number of clusters for partition P^t and X=$\bigcup_{l=1}^{k} C_l^t$, the number of clusters k can be different for different partitions. Define a similarity matrix $\widetilde{W}_{n\times n}$, where

$$\widetilde{W}_{ij} = \frac{1}{T}\sum_{t=1}^{T} W_{ij}(P^t), \tag{2}$$

$W_{ij}(P^t) = 1$, if $(i,j) \in C_k^t$, 0 otherwise, for some k.

```
• {Initialization:}
1. For all i ∈ 1, ...., n do
2.   $L_i = 0$;  $A_i = 0$;  $M_i = 0$;  $M_i^+ = 0$;
3. End for
• {Threshold learning:}
4. For all i ∈ 1, ...., n do
5.   Sample $k_t$ ants and compute max S(i, .) and < S(i, .) >;
     $T_i$ := (maxS(i, .)+ < S(i, .) >)/2
6. End for
• {Random meeting phase:}
7. For all k ∈ 1, ..., $k_m$ do
8.   Select a random pair (i, j), increase the age for both ants, compute S(i, j)
9.   Apply Rules 1-5
10. End for
• {Refine cluster:}
11. For all identified clusters do
      Compute $N_{cluster}$ and < $M^+$ > (averaged value of $M^+$ for all ants in the cluster)
12.   Compute the acceptance probability $P_\alpha = \alpha < M^+ > +(1-\alpha)\ N_{cluster}$
13.   If $P_\alpha < \theta_\gamma$ then Remove the cluster (all its parameters are reset to 0)
14. End for
15. For all ants having $M^+ < \theta_\alpha$ assign the ant to the cluster of the most similar
    ant j, which has a high enough acceptance degree ($M_j^+ > \theta_\alpha$)
• {Fuzzy C means part:}
16. Collect the unclassified data (the data with label $L_i = 0$ after the above
    steps)
17. The centroids formed after step 15 are taken as the initial cluster centers
    and the Fuzzy C means algorithm is applied to the unclassified data ($L_i = 0$)
18. Finally, merge newly labeled data with previously labeled data to form final
    partitions.
```

Fig. 1. ANTFCM algorithm with FCM used to refine the partition

Partition	X1	X2	X3	X4	X5	X6
I	1	1	1	2	2	1
II	2	1	1	2	2	2
III	2	1	2	1	2	2
IV	1	1	2	1	2	2
V	2	2	1	1	2	1

(a)

	X1	X2	X3	X4	X5	X6
X1		3/5	2/5	2/5	3/5	3/5
X2			2/5	2/5	1/5	1/5
X3				1/5	2/5	4/5
X4					2/5	2/5
X5						3/5
X6						

(b)

Fig. 2. (a) Example of cluster partitions: 6-examples data set with 5 partitions, the numbers 1 and 2 represent cluster labels. (b) Similarity matrix.

In Fig. 2 an example of five partitions of the data set and the similarity matrix are shown. We denote U as a solution of the consensus clustering problem. U is a connectivity matrix, if i, j belongs to the same cluster: $U_{ij} = 1$, otherwise it is 0. Therefore consensus clustering takes the form of the following optimization problem:

$$min_U \sum_{i,j=1}^{n} \left(\widetilde{W}_{ij} - U_{ij}\right)^2 = min_U \left\|\widetilde{W} - U\right\|^2 . \tag{3}$$

We used symmetric 3-factor NMF here, with the role of the connectivity matrix U (solution of NMF) and details of the algorithm in [12,13].

2.3 Determination of the Number of Clusters

In this section, we use the idea behind the model selection method in Brunet et al. [21] and examine its applicability to the NMF algorithm to determine the number of clusters in the data set. Brunet et al. successfully used this method to determine the unknown number of groups from gene expression data.

We first define the dispersion coefficient as

$$\rho_k = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} 4(\widehat{C}_k(i,j) - \frac{1}{2})^2 \tag{4}$$

where $\rho_k \in [0,1]$ and $\rho_k = 1$ represents a perfectly consistent assignment, k represents the number of clusters, the connectivity matrix $\widehat{C}_k$ is computed by averaging the connectivity matrices (U_k) over trials. The connectivity matrix U_k comes from the NMF solution (Eq. 3). The value $\widehat{C}_k(i,j)$ indicates the possibility of two examples i and j being assigned to the same cluster. If the assignment is consistent, each element of $\widehat{C}_k$ should be close to either 0 or 1. Therefore the dispersion coefficient (Eq. 4) is the summarization of the general quality of the consistency. After obtaining the values of ρ_k for various k's, the number of clusters can be determined by the point k where ρ_k is the maximum.

The procedure for finding the correct number of clusters is as follows:

1. Run the ANTFCM multiple times (we used 50 times here, and can reduce this number but too few will result in less accuracy), whatever the number of clusters ANTFCM finds, we take it as a candidate, if the candidate number is consistent all the time, we treat the candidate number as our solution.
2. For each candidate number of clusters, we applied the NMF algorithm with a different initialization over 5 trials based on the generated similarity matrix from the ANTFCM algorithm, using Eq. 4 we determined the correct number of clusters, **k** as the one where the ρ_k value is the maximum.

3 Experiments

3.1 Experimental Parameters

We first present the parameters needed by the ant clustering algorithm. We used the same default value for each parameter as in [20]. The $k_t = n/3$ parameter is used in the threshold learning phase to estimate the value of T_i for each ant, it is used for finding similarities between ant i and k_t randomly selected ants. $k_m = n^2$ represents times that an ant i randomly meets with other distinct ants during the random meeting phase. $\alpha = 0.1$ is included in the increasing and decreasing factors in Eq. 1. $\theta_\alpha = 0.1$ and $\theta_r = 0.1$ are used for the threshold values in the refining cluster stage. $m = 2$ is the degree of fuzzification for FCM.

3.2 Data Sets

The algorithm was applied to ten data sets. The Iris Plant database, Digits 389, Ionosphere, Letter IJL, Soybean, Wine Recognition data, Multiple Sclerosis, and Breast cancer are from the UCI data repository [17]. Digits 389 is a randomly sampled subset of three classes: 3, 8, and 9 from the Pen-Based Recognition of Handwritten Digits Data Set. Letter IJL is a randomly sampled subset of the letters I, J, and L from the letters dataset. Two artificial datasets were generated using Gaussian distributions. A mixture of five Gaussians was used. The first artificial data contains 1000 examples and 2 features, all classes are well separated. The 2nd dataset has 500 examples and 2 features but some examples are on the border among classes. Details are in [23].

3.3 Evaluation Metric for Clustering

Since all the above data sets come with labels, we can utilize the following accuracy measure:

$$Accuracy = (\sum_{C_k} max_{L_m} T(C_k, L_m))/n \tag{5}$$

where n is the number of examples, C_k is the k-th cluster, L_m is the m-th class and $T(C_k, L_m)$ is the number of examples that belong to class m assigned to cluster k. Thus accuracy is computed as the maximum sum of $T(C_k, L_m)$ for all pairs of clusters and classes.

3.4 Results for the ANTFCM Algorithm

The ANTFCM results in Table 1 are averaged over 50 experiments, each with a different random seed. The parameters k_t and k_m are critical parameters, k_t here is high enough to lead to a reliable estimation, k_m should be large enough to allow each ant to participate in meetings with other ants. The minimal value of k_m should be the number of ants, but in [8], a greater value proportional to n was suggested, we used n^2 here.

To show our ANTFCM methods provides a good data partition, we applied the classical Fuzzy C means (FCM) algorithm to the ten datasets. The accuracy for FCM is the average accuracy over 50 random initializations. Due to the different random seeds of each run, the ANTFCM algorithm may not find a consistent number of cluster centers and partitions. Therefore some ANTFCM accuracies higher than FCM result from over clustered partitions (Eq. 5). This instability also increased the difficulty in evaluating ANTFCM.

Table 1. Accuracy for ANTFCM and determined # of clusters. Entries in bold in the last column are where the # of clusters does not match the true number (in parentheses).

Data Name	FCM%	ANTFCM%	# of clusters from ANTFCM (candidate)	Maximal ρ_k	Determined # of clusters
Iris	89.33	81.20	3,4	0.7353	3
Digits 389	91.88	90.15	4	/	**4** (3)
Ionosphere	71.03	83.37	2,3,4	0.9175	2
Letter IJL	60.00	71.48	4,5,6,7	0.8197	**5** (3)
Soybean	78.72	95.06	3,4,5	0.9490	4
Wine	68.54	87.65	3	/	3
Multiple Sclerosis	83.67	83.33	3,4	0.8322	**3** (2)
Breast Cancer	95.28	94.28	2,3	0.7895	2
Artificial dataset 1	100.00	99.98	5	/	5
Artificial dataset 2	99.8	98.91	5	/	5

3.5 Determination of the Number of Clusters

From Table 1, we observe that for Digits 389, Wine and two artificial datasets a consistent number of clusters was found after ANTFCM. For the rest of data sets, we performed the two step procedure of Section 2.3. For most of the data sets the expected number of clusters was found except for Digits 389, Letter ILJ and Multiple Sclerosis. It is easy to find that these three datasets' candidate lists don't contain the correct number of clusters and they all overestimate the number of clusters. One possible way to solve this issue is to increase the value θ_r in the ant refining stage (line 13 in the ANTFCM algorithm), which can reduce the number of clusters generated, however, we are dealing with various datasets here and increasing θ_r may degrade performance for other datasets. A more adaptive method of choosing θ_r is needed.

3.6 Comparison with Consensus Results

We compared our NMF result with results from four other algorithms. In Table 2 KC represents the results of applying K-means to a consensus similarity matrix. CSPA is cluster-based similarity partitioning algorithm. HGPA is Hyper-Graph Partitioning Algorithm. A description of the CSPA and HGPA algorithms can be found in Strehl's paper [14]. Agglomerative represents the agglomerative clustering algorithm, which is a standard bottom-up algorithm, which can be stopped when the desired number of clusters is reached [22]. To fairly compare these five consensus algorithms, we used the known actual number of classes as an input for each algorithm (an advantage for them, though our combined method failed to detect the true number of classes for 3 data sets shown in Table 1, we can still use the known actual number of classes as an input for NMF here for fair accuracy comparison). Results were obtained by averaging over five trials with random initialization.

Table 2. Results on consensus clustering (50 runs of ANTFCM as input partitions)

Data Name	KC%	CSPA%	HPGA%	Agglomerative%	NMF%
Iris	81.33	92.67	92.67	80.67	94.00
Digits 389	85.75	92.32	49.34	90.79	90.79
Ionosphere	69.52	67.81	64.10	69.23	69.80
Letter IJL	60.29	53.33	54.76	60.00	59.52
Soybean	95.74	87.23	93.62	100.00	100.00
Wine	87.42	89.29	60.22	88.20	89.33
Multiple Sclerosis	78.57	70.40	74.49	70.40	70.40
Breast Cancer	94.71	83.40	65.52	94.71	94.71
Artificial dataset 1	97.36	92.30	24.80	100.00	100.00
Artificial dataset 2	89.12	94.00	25.20	99.8	99.8
Average accuracy	**83.98**	**82.28**	**60.47**	**85.38**	**86.83**

Table 3. Results on consensus clustering (50 runs of K-means as input partitions)

Data Name	KC%	CSPA%	HPGA%	Agglomerative%	NMF%
Iris	84.80	85.33	62.27	89.33	89.33
Digits 389	82.06	84.87	48.51	82.06	90.13
Ionosphere	71.23	67.81	64.10	71.23	71.23
Letter IJL	54.29	55.24	48.57	54.29	57.62
Soybean	71.06	72.34	78.72	78.72	78.72
Wine	70.22	67.98	64.49	70.79	70.22
Multiple Sclerosis	82.65	70.40	70.40	74.49	70.40
Breast Cancer	95.99	79.97	65.52	95.99	94.13
Artificial dataset 1	90.26	87.60	24.80	100.00	100.00
Artificial dataset 2	90.05	94.00	25.20	99.8	99.8
Average accuracy	**79.26**	**76.56**	**55.26**	**81.67**	**82.16**

From Table 2, we observed that NMF's average accuracy was the highest. This is a good indication that NMF can lead to a better quality solution. Interestingly, we observed that the Agglomerative algorithm's performance on the ten datasets was very close to the NMF based algorithm except for Iris and Wine.

In this paper, our approach was to combine an ant based algorithm and NMF together, therefore we need to compare the results with different input partitions. In Table 3, we used random initialization to obtain 50 K-means algorithm results. Comparing the results of the consensus algorithms between Table 2 and Table 3, we noticed that each algorithm using the result of ANTFCM as an input partition obtained better accuracy than when using the K-means algorithm. The reason behind this is that the ant based algorithm obtains final partitions by doing a form of global search. There is more chance for an ant based algorithm to find partitions which K-means or some other algorithms cannot find as they may get stuck in a local extrema when doing partitions. Hence, the promise of our new approach.

4 Summary

We introduced a new ant based clustering algorithm, in which an ensemble of data partitions was created followed by an NMF-based consensus clustering algorithm. We demonstrated the effectiveness of the two-stage algorithm of finding the correct number of clusters and improving the clustering performance by conducting a wide range of comparative experiments. The disadvantage of the first stage of our approach is that since the ants' process is self-organized, the final partition may not be the optimal solution. Therefore we utilized the 2nd stage, an NMF-based consensus clustering algorithm to find the optimal final partitions from an ensemble of ant generated partitions. The benefits of using NMF-based consensus clustering are: it can obtain a better partition; it can find a combined partition which is unattainable by any single clustering algorithm; it provides a consistent solution, which can be used to detect the correct number of clusters; finally it improves clustering robustness, which means the "mistakes" made by ANTFCM can be "canceled out" in the final consensus partition. In the case of the Iris data FCM found only one partition in over 10,000 random initializations. Our algorithm finds different partitions and when using NMF better reflects the underlying labeled data.

Acknowledgment

This research partially supported by the National Institutes of Health under grant number 1 R01 EB00822-01.

References

1. Hall, L.O., Ozyurt, I.B., Bezdek, J.C.: Clustering with a Genetically Optimized Approach. IEEE Transactions on Evolutionary Computation 3(2), 103–112 (1999)
2. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn., pp. 111–114. Prentice Hall, Upper Saddle River (2003)
3. Dorigo, M., Stutzle, T.: Ant Colony Optimization. Prentice Hall of India Private Limited, New Delhi (2005)
4. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Transaction on SMC-part B 26(1), 29–41 (1996)
5. Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A.: Detrain: The dynamics of collective sorting: robot-like ants and ant-like robots. In: Meyer, J.A., Wilson, S. (eds.) Proc. of the first Intern. Conf. on Simulation of Adaptive Behaviour: From Animals to Animats 1, pp. 356–365. MIT Press, Cambridge (1991)
6. Lumer, E., Faieta, B.: Diversity and Adaptation in Populations of Clustering Ants. In: Proc. of the third Interm. Conf. on Simulation of Adaptive Behavior: from Animals to Animats 3, pp. 501–508. MIT Press, Cambridge (1994)
7. Handl, J., Knowles, J., Dorigo, M.: Strategies for the Increased Robustness of Ant-based clustering. In: Di Marzo Serugendo, G., Karageorgos, A., Rana, O.F., Zambonelli, F. (eds.) ESOA 2003. LNCS, vol. 2977, pp. 90–104. Springer, Heidelberg (2004)

8. Labroche, N., Monmarche, N., Venturini, G.: A New Clustering Algorithm Based on the Chemical Recognition System of Ants. In: van Harmelen, F. (ed.) Proc. of the 15th European Conference on Artificial Intelligence, Lyon, France, pp. 345–349 (2002)
9. Fern, X.Z., Brodley, C.E.: Solving cluster ensemble problems by bipartite graph partitioning. In: ICML (2004)
10. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. In: ICDE, pp. 341–352 (2005)
11. Hu, X., Yoo, I., Zhang, X., Nanavati, P., Das, D.: Wavelet transformation and cluster ensemble for gene expression analysis. International Journal of Bioinformatics Research and Application 1(4), 447–460 (2006)
12. Li, T., Ding, C.: Weighted Consensus Clustering. In: Proceedings of the 2008 SIAM International Conference on Data Mining, Atlanta, April 24-26 (2008)
13. Ding, C., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix tri-factorizations for clustering. In: SIGKDD, pp. 126–135 (2006)
14. Strehl, A., Ghosh, J.: Cluster ensembles - a knowledge reuse framework for combining multiple partitions. Journal on Machine Learning Research (JMLR) 3, 583–617 (2002)
15. Lee, D.D., Seung, H.S.: Learning the parts of objects with nonnegative matrix factorization. Nature 401, 788–791 (1999)
16. Xu, W., et al.: Document Clustering Based on Non-Negative Matrix Factorization. In: SIGIR, pp. 267–273 (2003)
17. Blake, C.L., Newman, D.J., Hettich, S., Merz, C.J.: UCI repository of machine learning databases (1998)
18. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From natural to artificial systems. Oxford University Press, New York (1999)
19. Hall, L.O., Kanade, P.M.: Swarm Based Fuzzy Clustering with partition Validity. In: FUZZ 2005. The 14th IEEE International Conference on Fuzzy Systems, 2005, May 22-25, pp. 991–995 (2005)
20. Zaharie, D., Zamfirache, F.: Dealing with noise in ant-based clustering. In: Proc. IEEE Congress of Evolutionary Computation 2005, Edinburgh, pp. 2395–2402 (2005)
21. Brunet, J., Tamayo, P., Golub, T., Mesirov, J.: Metagenes and molecular pattern discovery using matrix factorization. Proceedings of the National Academy of Sciences 101(12), 4164–4169 (2004)
22. Kurita, T.: An efficient agglomerative clustering algorithm for region growing. In: Proc. of IAPR Workshop on Machine Vision Applications, pp. 210–213 (1994)
23. Gu, Y., Hall, L.O., Goldgof, D.B.: Ant Clustering with consensus, Technical Report ISL-1-2009, http://www.cse.usf.edu/~hall/papers/tr109.pdf

Selective Ensemble under Regularization Framework

Nan Li and Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210093, China
{lin,zhouzh}@lamda.nju.edu.cn

Abstract. An ensemble is generated by training multiple component learners for a same task and then combining them for predictions. It is known that when lots of trained learners are available, it is better to ensemble some instead of all of them. The selection, however, is generally difficult and heuristics are often used. In this paper, we investigate the problem under the regularization framework, and propose a regularized selective ensemble algorithm RSE. In RSE, the selection is reduced to a quadratic programming problem, which has a sparse solution and can be solved efficiently. Since it naturally fits the semi-supervised learning setting, RSE can also exploit unlabeled data to improve the performance. Experimental results show that RSE can generate ensembles with small size but strong generalization ability.

1 Introduction

Ensemble learning [11] is a learning paradigm where multiple component learners are trained to solve a problem. Since an ensemble often has better performance than a single learner, it has achieved successes in many domains.

In general, an ensemble is built in two steps, i.e., training multiple component learners and then combining them. According to the styles of training component learners, popular ensemble algorithms can be roughly categorized into two classes, that is, approaches where component learners are trained in parallel, and approaches where component learners must be trained sequentially. Representatives of the former include Bagging [4], Random Subspace [14], Random Forest [6], GASEN [24], etc. Representatives of the latter include AdaBoost [13], Arc-x4 [5], LPBoot [10], etc.

In most ensemble algorithms, all of the obtained component learners are employed to build an ensemble. However, some researchers [24,23,17] show that through *selective ensemble*, i.e., ensembling some instead of all the available component learners, a better ensemble can be generated. The selection, however, is not easy and thus many heuristics have been used. For example, for the selection, Zhou et al. [24,23] used a genetic algorithm; Castro et al. [7] employed artificial immune algorithm; Coyle and Smith [9] utilized case similarity; Martínez-Muñoz and Suárez [17] proposed to order the component learners and then select the first ones to use.

As a generic learning framework, regularized approaches [18] work by minimizing the regularized risk function, and have been found useful in ensemble learning. For example, LPBoost [10] takes margin as regularizer; RegBoost [15] utilizes the graph Laplacian regularizer [2] to make base classifiers cut through sparse regions; in [8], the regularizer is used to take local smoothness constraints, yielding a more general regularized Boosting.

In contrast to previous heuristic methods, in this paper we study the selective ensemble problem under the regularization framework. We utilize the hinge loss and the graph Laplacian regularizer, and present a regularized selective ensemble approach RSE. In RSE, the selection problem is reduced to a quadratic programming (QP) problem, which has a sparse solution and meanwhile can be solved efficiently. Empirical study shows that RSE is able to generate ensembles with small size while have strong generalization ability, which is superior to many existed ensemble methods.

In practical applications, unlabeled examples are much easier to obtain. Therefore, semi-supervised learning [25], which attempts to exploit unlabeled examples to help improve learning performance, has attracted much attention. A prominent advantage of RSE is that it naturally fits the semi-supervised setting. Experiments show that RSE can exploit unlabeled data effectively.

In the following of the paper we will start with a brief review on related work. Then, we propose RSE and its semi-supervised extension, followed by reports on experiments. Finally, we conclude the paper.

2 Related Work

After obtaining the component learners, most ensemble algorithms combine all of them to build an ensemble, however, it has been shown that it is better to ensemble some instead of all of them [24,23,17].

Zhou et al. [24] analyzed the relationship between ensemble and its component learners from the context of both regression and classification, and proved that it may be better to combine many instead of all of the learners. Since it is difficult to select the component learners, they used genetic algorithm to select a part of learners to build the ensemble. Empirical studies show that, comparing with some popular ensemble approaches such as Bagging and Boosting, the proposed GASEN algorithm can generate ensembles with smaller sizes but stronger generalization ability.

Martínez-Muñoz and Suárez [17] proposed a heuristic method, where the component learners obtained from Bagging are reordered, and a part of the top-ranked ones are included in the final ensemble. The experimental result also shows that selective ensemble may improve ensemble's performance while reducing its size.

Selective ensemble is a special paradigm of ensemble learning, which aims at building strong ensembles with small sizes. In some sense it is related to ensemble pruning [16,22]. However, it is noteworthy that in earlier researches of ensemble prunning [16], the goal was to use a small size of ensemble to achieve an equivalent performance of a boosted ensemble. This has been proved to be NP-hard and is even hard to approximate [19], and the pruning may sacrifice the generalization ability of the final ensemble. Since Zhou et al.'s work [24], it is known that by using selective ensemble it is possible to get a small yet strong ensemble. What makes this difference is that earlier ensemble pruning works on ensembles where the component learners must be trained sequentially, while selective ensemble works on ensembles where the component learners can be generated in parallel, and the sequential generation of the former makes the problem much more difficult to tackle.

3 The RSE Approach

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the feature space and the set of class labels, respectively. Here, binary classification is considered for simplicity, assuming $\mathcal{Y} = \{-1, +1\}$. A training set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ is given, where $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$.

From training set $\mathcal{D}$, ensemble learning algorithms try to train a set of component classifiers $\{h_1, \ldots, h_M\}$ and choose weights $\{w_1, \ldots, w_M\}$ to combine them as $H(\boldsymbol{x}) = \sum_{i=1}^{M} w_i h_i(\boldsymbol{x})$, and it is often assumed that $w_i \geq 0$ and $\sum_{i=1}^{M} w_i = 1$. The classification decision of the ensemble H on instance $\boldsymbol{x}$ is $+1$ if $H(\boldsymbol{x}) \geq 0$ and -1 otherwise. The number of component classifiers is called the *size* of the ensemble. It is obvious that classifiers with zero weights will be excluded from the ensemble.

In this work, we concern on the second step of building an ensemble, i.e., choosing weights to combine the component classifiers. In this section, we study the selective ensemble problem under the regularization framework.

3.1 Regularized Risk Function

Let $\boldsymbol{w} = [w_1, \ldots, w_M]^\top$ denote the weights used to combine the component classifiers $\{h_1, \ldots, h_M\}$. The ensemble's output on instance $\boldsymbol{x}_i$ is

$$H(\boldsymbol{x}_i) = \sum\nolimits_{k=1}^{M} w_k h_k(\boldsymbol{x}_i) = \boldsymbol{p}_i^\top \boldsymbol{w}, \tag{1}$$

where $\boldsymbol{p}_i = [h_1(\boldsymbol{x}_i), \ldots, h_M(\boldsymbol{x}_i)]^\top$ are the component classifiers' predictions on $\boldsymbol{x}_i$.

Under the regularization framework, the weights $\boldsymbol{w}$ is usually determined by minimizing the regularized risk function

$$R(\boldsymbol{w}) = \lambda\, V(\boldsymbol{w}) + \Omega(\boldsymbol{w}), \tag{2}$$

where $V(\boldsymbol{w})$ is the empirical loss which approximately measures the misclassification loss of classifier on the training examples in $\mathcal{D}$, $\Omega(\boldsymbol{w})$ is the regularization term, and λ is the regularization parameter which specifies the tradeoff between the minimization of $V(\boldsymbol{w})$ and the smoothness or simplicity enforced by minimization of $\Omega(\boldsymbol{w})$.

Empirical Loss. The empirical loss considers the loss between each example's class label y_i and its corresponding prediction $H(\boldsymbol{x}_i) = \boldsymbol{p}_i^\top \boldsymbol{w}$.

Here, we use the *hinge loss* function $\ell\big(y_i, H(\boldsymbol{x}_i)\big) = \max\big(0, 1 - y_i H(\boldsymbol{x}_i)\big)$. Typically, this is the empirical loss minimized in support vector machines [18], and it is usually written as a sum of slack variables included in the constraints. Afterwards, we can define the empirical loss function $V(\boldsymbol{w})$ for the ensemble H on training set $\mathcal{D}$ as

$$V(\boldsymbol{w}) = \sum\nolimits_{i=1}^{N} \ell\big(y_i, H(\boldsymbol{x}_i)\big) = \sum\nolimits_{i=1}^{N} \max(0, 1 - y_i \boldsymbol{p}_i^\top \boldsymbol{w}), \tag{3}$$

where, as same as previous definition, $\boldsymbol{w}$ are the weights, and $\boldsymbol{p}_i$ are predictions on $\boldsymbol{x}_i$. Obviously, the empirical loss function in Eq. 3 is convex and continuous.

Regularization Term. The regularization term $\Omega(\boldsymbol{w})$ is used to smooth or simplify the function. In this work, the *graph Laplacian regularizer* [2] is adopted.

Let $G = (\mathcal{V}; \mathcal{E})$ be the neighborhood graph of the training set $\mathcal{D}$, where the vertex set $\mathcal{V}$ is the set of all the examples in $\mathcal{D}$, and the edge set $\mathcal{E}$ contains pairs of neighboring examples. Based on the idea that the classifier should make similar predictions on neighboring examples, the classifier that cuts through dense regions is penalized by

$$P_{\mathcal{L}}(H) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} W_{ij} \big(H(\boldsymbol{x}_i) - H(\boldsymbol{x}_j)\big)^2, \tag{4}$$

where $\boldsymbol{W}$ is the (weighted) adjacency matrix of G.

Let the diagonal matrix $\boldsymbol{D}$ defined by $D_{ii} = \sum_{j=1}^{N} W_{ij}$, the normalized *graph Laplacian* of G is $\boldsymbol{L} = \boldsymbol{D}^{-1/2}(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{D}^{-1/2}$. Then, the function in Eq. 4 can be rewritten as $P_{\mathcal{L}}(H) = \boldsymbol{h}^\top \boldsymbol{L} \boldsymbol{h}$, where $\boldsymbol{h} = [H(\boldsymbol{x}_1), \ldots, H(\boldsymbol{x}_N)]^\top$ is ensemble H's predictions on the examples. By Eq. 1, it follows that $\boldsymbol{h} = [\boldsymbol{p}_1^\top \boldsymbol{w}, \ldots, \boldsymbol{p}_N^\top \boldsymbol{w}] = \boldsymbol{P}^\top \boldsymbol{w}$, where $\boldsymbol{P} \in \{-1, +1\}^{M \times N}$ is the *prediction matrix* which collects every component classifier's prediction on every example, and $\boldsymbol{P}_{ij} = h_i(\boldsymbol{x}_j)$.

Using the graph Laplacian regularizer, the regularization term is defined as

$$\Omega(\boldsymbol{w}) = \boldsymbol{h}^\top \boldsymbol{L} \boldsymbol{h} = \boldsymbol{w}^\top \boldsymbol{P} \boldsymbol{L} \boldsymbol{P}^\top \boldsymbol{w}, \tag{5}$$

where, as defined above, $\boldsymbol{L}$ is the *graph Laplacian* and $\boldsymbol{P}$ is the *prediction matrix*.

Here, rather than building the neighbor graph on the training set, we assume that the graph is fully connected, and the weight of the edge between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is determined by the distance function $W_{ij} = \exp(-\sigma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$, where σ is a bandwidth parameter.

3.2 Optimization Problem with Sparsity Constraint

Using the empirical loss function defined by Eq. 3 and the graph Laplacian regularization term in Eq. 5, we have the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \boldsymbol{w}^\top \boldsymbol{P} \boldsymbol{L} \boldsymbol{P}^\top \boldsymbol{w} + \lambda \sum_{i=1}^{N} \max(0, 1 - y_i \boldsymbol{p}_i^\top \boldsymbol{w}) \\ \text{subject to} \quad & \boldsymbol{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \boldsymbol{0}. \end{aligned} \tag{6}$$

Since $\max(\cdot, \cdot)$ is non-smooth, Eq. 6 can be equivalently rewritten as the following

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \boldsymbol{w}^\top \boldsymbol{P} \boldsymbol{L} \boldsymbol{P}^\top \boldsymbol{w} + \lambda \, \boldsymbol{1}^\top \boldsymbol{\xi} \\ \text{subject to} \quad & y_i \boldsymbol{p}_i^\top \boldsymbol{w} + \xi_i \geq 1, \quad i = 1, \ldots, N \\ & \boldsymbol{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \boldsymbol{0}, \quad \boldsymbol{\xi} \geq \boldsymbol{0} \end{aligned} \tag{7}$$

where $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_N]^\top$ are slack variables, λ is the regularization parameter, $\boldsymbol{1}$ and $\boldsymbol{0}$ are all-1 and all-0 vector, respectively. It is evident that Eq. 7 is a standard QP problem, which can be efficiently solved using many existed optimization packages.

Note that the constraint, $\boldsymbol{1}^\top \boldsymbol{w} = 1, \boldsymbol{w} \geq \boldsymbol{0}$, is ℓ_1-norm constraint on the weights $\boldsymbol{w}$, which is a sparsity constraint that will force some w_i's to be zero. Recall that the ensemble size is equal to the number of non-zero elements in $\boldsymbol{w}$, thus ensemble with small size is encouraged.

```
Input:   training set $\mathcal{D}$, component learner $L$, trials $M$
Process:
  1.  for $i = 1$ to $M${
  2.          $\mathcal{S}_i$ = bootstrap sample from $\mathcal{D}$ ;
  3.          $h_i = L(\mathcal{S}_i)$ ;   }
  4.  Get the component classifiers' predictions $\boldsymbol{P}$;
  5.  Build the neighbor graph on $\mathcal{D}$, and calculate $\boldsymbol{L}$;
  6.  Solve the QP problem in Eq. 7 to get the weights $\boldsymbol{w}$;
Output:  ensemble $H$
  $H(\boldsymbol{x}) = \sum_{w_i>0} h_i(\boldsymbol{x})$        for selective ensemble (RSE)
  $H(\boldsymbol{x}) = \sum_{w_i>0} w_i h_i(\boldsymbol{x})$    for weighted ensemble (RSE-w)
```

Fig. 1. Pseudo-code of the RSE Algorithm

3.3 The RSE Algorithm

The RSE algorithm is summarized in Fig. 1.

Note that RSE uses bootstrap sampling to train component classifiers, which is borrowed from Bagging, other methods to train component classifiers can also be used.

3.4 Semi-supervised Extension

In many real-world applications, it is often the case that abundant unlabeled examples are available. Thus, semi-supervised learning [25], which attempts to exploit unlabeled examples to improve the performance, has attracted much attentions.

Using the unlabeled examples for regularization [2] is one of the important approaches in semi-supervised learning, and *graph Laplacian regularizer* is a representative example. RSE uses the graph Laplacian regularizer, and it naturally fits the semi-supervised learning setting, we call the corresponding semi-supervised version as RSE$_{ss}$, and RSE-w$_{ss}$ corresponds to the semi-supervised version of RSE-w.

Obviously, the graph Laplacian regularizer does not rely on the class labels. So, when given training set $\mathcal{D}$ with L labeled examples and U unlabeled ones, the graph Laplacian regularizer can be derived by Eq. 5, except that the prediction matrix $\boldsymbol{P}$ and the graph Laplacian $\boldsymbol{L}$ should be computed on both labeled and unlabeled examples.

4 Experiments

4.1 Configuration

We use 14 two-classes UCI data sets [3] in experiments. These data sets span a broad range of real domains, and some statistics of the data sets are shown in Table 1.

We compared RSE with Bagging [4], AdaBoost [13], and the first selective ensemble algorithm GASEN [24]. RSE-w is also evaluated. Without loss of generality, C4.5 tree is used as the component classifier. For Bagging and AdaBoost, the ensemble size is 100, while GASEN and RSE select component learners from 100 bagged C4.5 trees.

Table 1. Experimental data sets

Date set	#Examples	#Attributes	Date set	#Examples	#Attributes
australian	690	14	*heart-s*	270	13
ballons	76	4	*ionosphere*	351	34
cancer	286	9	*kr-vs-kp*	3196	36
cylinder-b	540	39	*live-dis*	345	6
diabetes	768	8	*spectf*	267	44
germen	1000	20	*spambase*	4601	57
haberman	306	3	*vote*	435	16

Table 2. Comparison of the predictive errors (mean±std) under supervised setting, where the best performance on each data set is bolded. The *average* row presents the results averaged over all data sets; *W/T/L* row summarizes the comparison of RSE (RSE-w) against other algorithms according to pairwise t-tests with 95% significance level.

Data set	C4.5	Bagging	AdaBoost	GASEN	RSE-w	RSE
australian	.148±.021	.133±.018	.137±.015	.131±.017	.131±.018	**.127**±.016
balloons	.316±.067	.269±.089	.243±.077	.262±.079	**.223**±.069	.241±.074
cancer	.278±.035	.267±.027	.343±.049	.273±.029	.272±.020	**.266**±.018
cylinder-b	.328±.021	.328±.021	.328±.020	.322±.020	.337±.045	**.319**±.033
diabetes	.267±.024	**.241**±.024	.265±.023	**.241**±.021	.246±.018	.243±.019
germen	.281±.018	.249±.019	.258±.025	.248±.019	.249±.017	**.245**±.017
haberman	.285±.040	.266±.031	.308±.036	.267±.029	.260±.013	**.256**±.020
heart-s	.217±.041	.181±.034	.188±.029	.177±.031	.175±.030	**.168**±.031
ionosphere	.110±.030	.077±.023	**.062**±.023	.075±.022	.074±.021	.070±.021
kr-vs-kp	.008±.003	.007±.003	**.005**±.003	.007±.002	.006±.002	.006±.002
liver-dis	.352±.036	.283±.035	.317±.046	.283±.033	.276±.037	**.269**±.034
spectf	.173±.036	.125±.028	.128±.029	.125±.028	.129±.027	**.122**±.028
spambase	.079±.008	.060±.007	**.050**±.006	.059±.007	.059±.007	.059±.007
vote	.047±.016	.042±.015	.056±.017	.042±.014	.040±.012	**.038**±.012
average	0.206	0.183	0.189	0.179	0.176	**0.171**
W/T/L (RSE)	—	11/3/0	9/3/2	6/8/0	8/5/1	—
W/T/L (RSE-w)	—	3/11/0	9/2/3	3/9/2	—	1/5/8

They are all implemented in WEKA [21], and the trees are pruned following the default settings. The genetic algorithm employed by GASEN is implemented using MATLAB [12], the selection threshold is set to 0.01, and the parameters of genetic algorithm are set to the default values. The bandwidth σ is set to 0.01, and the regularization parameter λ is selected by 5-fold cross validation on training sets for RSE and RSE-w, respectively. The QP problem is solved using MOSEK [1].

For each data set, 50 runs of hold-out tests are executed. In each run, one-third of examples are selected randomly for testing, and the remaining for training. The average predictive error rates and the ensemble sizes are recorded.

In order to study how well RSE$_{ss}$ and RSE-w$_{ss}$ can exploit unlabeled examples, experiments under semi-supervised setting are also carried out, during which the training data set is partitioned into labeled set $\mathcal{L}$ and unlabeled set $\mathcal{U}$ under a *label rate*. To simulate different amount of unlabeled examples, two different label rates, 5% and 10%, are investigated here. Under 5% (10%) label rate, 5% (10%) of the training set is used as labeled training data, while the remaining 95% (90%) as unlabeled data.

Table 3. Comparison of the ensemble size (mean±std) under supervised setting, where the smallest size on each data set is bolded. The *average* row presents the size averaged over all data sets.

Date set	GASEN	RSE-w	RSE
australian	45.3±3.0	20.5±5.7	**19.1**±5.7
balloons	38.9±**15.8**	15.8±14.6	15.9±18.9
cancer	41.0±4.2	**11.9**±3.6	13.0±4.2
cylinder-b	15.5±13.7	**9.1**±6.7	11.4±9.5
diabetes	45.0±2.5	**26.5**±7.2	**26.5**±8.2
germen	45.5±2.8	29.9±5.0	**29.3**±4.5
haberman	43.1±9.1	15.1±4.8	**13.3**±4.7
heart-s	45.5±3.1	**19.8**±7.7	21.7±7.2
ionosphere	43.1±8.7	15.0±6.3	**14.9**±5.6
kr-vs-kp	40.4±7.1	10.1±6.3	**8.9**±5.8
liver-dis	44.7±3.4	31.4±8.6	**29.9**±8.3
spectf	45.0±3.3	17.7±4.4	**17.3**±3.4
spambase	44.5±3.3	37.8±5.6	**37.2**±5.8
vote	38.6±16.1	8.8±6.0	**8.5**±5.8
average	41.2	19.2	**19.1**

Table 4. Comparison of the predictive errors (mean±std) under 5% label rate, where the best one on each data set is bolded. The *average* row presents the results averaged over all data sets.

Data set	C4.5	Bagging	AdaBoost	GASEN	RSE-w	RSE	RSE-w$_{ss}$	RSE$_{ss}$
australian	.200±.065	.184±.051	.210±.050	.206±.061	.201±.053	.197±.051	**.151**±.025	**.151**±.023
balloons	**.458**±.001	**.458**±.001	**.458**±.001	.475±.033	**.458**±.001	**.458**±.001	**.458**±.001	**.458**±.001
cancer	.312±.062	.310±.051	.336±.074	.340±.091	.289±.016	.288±.023	.292±.010	**.285**±.022
cylinder-b	.394±.060	.393±.054	.395±.046	.396±.053	.403±.064	.420±.061	.336±.038	**.332**±.037
diabetes	.342±.046	.299±.036	.317±.035	.302±.038	.289±.035	.286±.034	.262±.027	**.259**±.028
germen	.345±.044	.308±.028	.326±.032	.310±.028	.329±.033	.316±.032	.291±.012	**.284**±.015
haberman	.280±.055	.273±.041	.288±.056	.292±.059	.257±.012	.265±.030	.256±.012	**.255**±.015
heart-s	.318±.081	.268±.070	.302±.079	.287±.078	.296±.069	.291±.071	**.220**±.045	.221±.043
ionosphere	.267±.095	.238±.068	.258±.084	.242±.073	.226±.065	.218±.056	**.176**±.030	**.176**±.029
kr-vs-kp	.074±.021	.064±.016	.067±.017	.064±.015	.065±.016	.064±.016	**.047**±.013	.049±.014
liver-dis	.443±.059	.434±.047	.435±.054	.427±.052	.411±.028	.404±.038	.379±.037	**.375**±.040
spectf	.362±.082	.281±.055	.315±.065	.300±.059	.328±.067	.336±.072	.262±.019	**.245**±.030
spambase	.147±.019	.107±.015	.087±.013	.107±.014	.116±.015	.110±.011	.100±.011	**.098**±.011
vote	.097±.068	.094±.062	.103±.072	.087±.058	.096±.061	.084±.058	**.059**±.045	.060±.046
average	0.288	0.265	0.278	0.274	0.269	0.267	0.235	**0.232**

4.2 Results

Results under Supervised Setting. The predictive errors are shown in Table 2, the last rows present the average error over all data sets, and the summary of pairwise t-tests with 95% significance level, where *W/T/L* means that RSE (RSE-w) wins, ties and loses on *#W*, *#T* and *#L* data sets, respectively. The ensemble sizes are presented in Table 3.

It can be observed from Table 2 that selective ensemble algorithms consistently perform better than single decision trees. Moreover, RSE has the lowest averaged error rate, and t-test results indicate that RSE performs significantly better than all the comparing methods. Compared with the non-selective ensemble algorithms Bagging and AdaBoost, RSE wins on 11 and 9 data sets. respectively. RSE only loses to AdaBoost on *ionosphere* and *spambase*, where AdaBoost performs significantly better and others ensemble methods have similar performance. Compared with GASEN, RSE also performs quite well (wins on 6 data sets and never loses).

It is quite interesting that RSE performs much better than RSE-w which weights the selected classifier by $\boldsymbol{w}$. Similar phenomenon was observed before for GASEN [24]. A possible reason is that the weights of RSE-w were determined by minimizing the hinge

Table 5. Comparison of the predictive errors (mean±std) under 10% label rate, where the best one on each data set is bolded. The *average* row presents the results averaged over all data sets.

Data set	C4.5	Bagging	AdaBoost	GASEN	RSE-w	RSE	RSE-w$_{ss}$	RSE$_{ss}$
australian	.182±.041	.167±.032	.190±.037	.173±.029	.191±.035	.177±.036	.152±.016	**.149**±.019
balloons	.423±.114	.403±.099	.425±.100	.405±.113	.374±.079	.390±.107	.368±.070	**.346**±.073
cancer	.297±.029	.293±.034	.333±.047	.299±.033	.289±.015	.283±.022	.280±.025	**.275**±.026
cylinder-b	.369±.055	.349±.048	.377±.057	.347±.044	.370±.055	.380±.052	.318±.029	**.306**±.025
diabetes	.310±.046	.279±.038	.301±.045	.277±.039	.272±.031	.262±.034	.250±.026	**.243**±.025
germen	.335±.036	.300±.024	.309±.029	.299±.024	.305±.028	.289±.019	.286±.017	**.277**±.016
haberman	.273±.060	.267±.033	.288±.054	.283±.052	**.249**±.018	.254±.019	.251±.017	**.249**±.023
heart-s	.290±.058	.251±.063	.266±.066	.255±.073	.267±.058	.246±.052	**.212**±.039	**.212**±.046
ionosphere	.203±.083	.190±.053	.199±.078	.190±.058	.181±.054	.173±.052	.148±.039	**.143**±.035
kr-vs-kp	.057±.015	.051±.010	.042±.012	.049±.010	.044±.013	.044±.012	**.035**±.008	.036±.009
liver-dis	.403±.053	.397±.058	.398±.058	.397±.057	.384±.046	.378±.041	.347±.035	**.346**±.036
spectf	.328±.067	.269±.043	.289±.066	.273±.045	.284±.045	.285±.054	.235±.025	**.226**±.031
spambase	.127±.012	.095±.011	.073±.007	.097±.012	.099±.011	.094±.010	.088±.009	**.087**±.009
vote	.074±.049	.058±.029	.080±.044	.069±.038	.072±.033	.061±.029	**.049**±.023	**.049**±.023
average	0.262	0.241	0.255	0.244	0.241	0.237	0.216	**0.210**

loss, which is only an approximation of the 0-1 misclassification loss, while the optimal weights for the hinge loss may not be optimal for the 0-1 loss.

It can be found from Table 3 that the ensemble sizes of RSE and RSE-w are much smaller than that of GASEN. Since different regularization parameters are selected via cross validation, RSE and RSE-w select 19.1 and 19.2 trees, respectively, while GASEN selects 41.2 trees on average.

Based on these result, it is believed that RSE can generate ensembles with small size but strong generalization ability.

Results under Semi-Supervised Setting. The predictive error rates under label rates of 5% and 10% are shown in Table 4 and Table 5, respectively, and the ensemble sizes are shown in Table 6 and Table 7, respectively. Here, RSE$_{ss}$ and RSE-w$_{ss}$ use unlabeled data to improve performance, while other methods only use the labeled data.

It can be found from Table 4 and Table 5 that, under 5% and 10% label rates, RSE$_{ss}$ get the best performance on 11 and 13 data sets, respectively. Comparing RSE$_{ss}$ with RSE under 5% label rate, it can be found that the averaged error rate is reduced from 0.237 to 0.210. Especially, it is reduced from 0.336 to 0.245 on *spectf*. On *cylinder-b*, RSE has the worst error rate 0.420, but RSE$_{ss}$ performs best with error rate 0.332. Similar results can be found by comparing RSE-w$_{ss}$ with RSE-w, also under 10% label rate. Therefore, we can see the unlabeled data could be useful in constructing ensemble, even when the component classifiers are learned on the labeled data.

It can be found that under 5% label rate, the average error rate of RSE is 0.267 and that of RSE$_{ss}$ is 0.232, with a reduction of 15.1%; while under 10% label rate, the average error rate of RSE is 0.237 and that of RSE$_{ss}$ is 0.210, with a reduction of 12.9%. This validates that RSE$_{ss}$ can benefit from using unlabeled examples.

Comparing ensemble sizes in Table 6 and Table 7, we can find that RSE$_{ss}$ also generates smaller ensembles than GASEN, and much smaller than that of Bagging and AdaBoost. The sizes of RSE$_{ss}$ (RSE-w$_{ss}$) ensembles are often larger than that of RSE(RSE-w) ensembles; this is not difficult to understand because RSE$_{ss}$(RSE-w$_{ss}$) uses much more data than RSE(RSE-w) since the latter does not use unlabeled data.

Table 6. Comparison of the ensemble size (mean±std) under 5% label rate, where the smallest size on each data set is bolded. The *average* row presents the size averaged over all data sets.

Data set	GASEN	RSE-w	RSE	RSE-w$_{ss}$	RSE$_{ss}$
australian	34.9±9.5	**10.1**±3.8	11.6±3.5	25.1±9.1	30.4±8.3
balloons	**13.4**±9.7	75.2±4.5	75.2±4.5	75.2±4.5	75.2±4.5
cancer	**27.7**±9.6	53.1±18.2	53.6±14.4	50.2±15.0	50.7±17.9
cylinder-b	46.9±15.0	**4.0**±12.3	4.1±4.4	8.5±3.8	12.4±4.2
diabetes	46.9±13.6	**7.0**±3.7	7.8±7.2	22.0±9.7	21.0±7.2
germen	49.5±6.1	5.5±2.6	**5.3**±2.7	8.0±4.9	10.5±4.4
haberman	**25.7**±12.3	65.8±13.9	65.2±14.4	71.3±20.8	63.0±21.4
heart-s	**29.5**±14.5	36.1±10.9	37.6±13.1	49.3±13.2	49.1±15.5
ionosphere	34.3±14.1	36.7±14.8	48.2±19.6	**13.3**±8.0	19.2±9.2
kr-vs-kp	41.2±10.3	10.9±6.7	10.7±8.2	**7.1**±8.4	7.9±8.0
liver-dis	47.7±11.7	20.2±16.3	**19.5**±11.7	20.2±7.0	23.4±11.2
spectf	38.8±11.4	27.6±14.7	27.9±14.9	**17.3**±9.4	25.7±11.2
spambase	50.4±4.8	**10.6**±3.0	11.3± 2.5	17.8±5.9	18.5±4.3
vote	**24.6**±9.8	58.7±19.5	64.5±17.7	63.6±19.6	70.8±19.6
average	36.5	30.1	31.6	32.3	34.1

Table 7. Comparison of the ensemble size (mean±std) under 10% label rate, where the smallest size are bolded. The *average* row presents the size averaged over all data sets.

Data set	GASEN	RSE-w	RSE	RSE-w$_{ss}$	RSE$_{ss}$
australian	39.2±5.9	**4.9**±3.9	5.1±4.6	6.8±3.1	8.4±4.8
balloons	**29.7**±13.9	48.0±17.9	61.6±15.8	39.9±11.1	65.6±17.3
cancer	**33.8**±11.3	37.2±13.3	42.3±13.3	25.2±15.6	37.0±12.8
cylinder-b	44.6±7.8	**2.7**±1.6	2.9±1.7	8.3±8.6	10.1±8.8
diabetes	44.8±5.4	**7.6**±3.8	8.4±3.6	15.8±10.5	11.9±11.3
germen	44.9±3.6	6.9±3.7	8.3±3.6	**6.3**±6.3	11.3±8.6
haberman	**27.4**±6.1	52.1±9.6	56.4±7.4	53.2±9.7	55.7±12.9
heart-statlog	42.3±14.3	12.4±17.7	**10.8**±15.0	18.2±9.4	17.5±11.7
ionosphere	35.5±20.5	15.0±9.1	15.8±10.9	7.6±16.3	**11.9**±11.6
kr-vs-kp	41.4±13.9	9.7±7.0	8.9±6.5	8.9±8.1	**6.2**±7.9
liver-dis	42.7±13.5	**7.7**±4.2	8.2±4.6	8.1±6.8	12.0±9.0
spectf	43.6±11.9	7.3±8.2	**6.9**±6.5	7.3±6.8	10.2±9.9
spambase	44.1±9.0	**14.0**±3.4	14.4±2.7	21.2±5.0	21.6±4.4
vote	**15.3**±12.8	57.5±21.1	66.4±16.9	57.3±14.1	64.7±13.7
average	37.8	20.2	22.6	20.3	24.6

5 Conclusion

Given a number of trained component learners, it is better to build an ensemble that contains some instead of all of the component learners. The selection, however, is generally difficult and some smart heuristics were used in previous studies. In this paper, we study the problem of selective ensemble under the regularization framework. In the proposed RSE approach, the selection problem is reduced to a QP problem which has a sparse solution and can be efficiently solved. Moreover, RSE can exploit unlabeled examples easily. Experiments show that RSE can generate ensembles with smaller sizes but strong generalization ability, and the use of unlabeled data is helpful.

In this paper, RSE is designed to handle two-class classification problems. A multi-class extension will be studied in future work. Considering that combining class probabilities may lead to better results than combining class labels [20], it may also an interesting future work is to develop selective ensemble algorithms which selectively combine class probabilities.

Acknowledgment

This work was supported by the National Science Foundation of China (60635030, 60721002) and the Jiangsu Science Foundation (BK2008018).

References

1. Andersen, E.D., Jensen, B., Sandvik, R., Worsoe, U.: The improvements in mosek version 5. Technical report, The MOSEK Inc. (2007)
2. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of Machine Learning Research 7, 2399–2434 (2006)
3. Blake, C., Keogh, E., Merz, C.J.: UCI repository of machine learning databases (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
4. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
5. Breiman, L.: Arcing classifiers. Annals of Statistics 26(3), 801–849 (1998)
6. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
7. Castro, P.D., Coelho, G.P., Caetano, M.F., Von Zuben, F.J.: Designing ensembles of fuzzy classification systems: An immune-inspired approach. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 469–482. Springer, Heidelberg (2005)
8. Chen, K., Wang, S.: Regularized boost for semi-supervised learning. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems, vol. 20, pp. 281–288. MIT Press, Cambridge (2008)
9. Coyle, M., Smyth, B.: On the use of selective ensembles for relevance classification in case-based web search. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS, vol. 4106, pp. 370–384. Springer, Heidelberg (2006)
10. Demiriz, A., Bennett, K.P., Shawe-Taylor, J.: Linear programming boosting via column generation. Machine Learning 46(1-3), 225–254 (2006)
11. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
12. Doherty, D., Freeman, M.A., Kumar, R.: Optimization with matlab and the genetic algorithm and direct search toolbox. Technical report, The MathWorks Inc. (2004)
13. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
14. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transaction on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
15. Kégl, B., Wang, L.: Boosting on manifolds: Adaptive regularization of base classifiers. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 17, pp. 665–672. MIT Press, Cambridge (2005)
16. Margineantu, D., Dietterich, T.G.: Pruning adaptive boosting. In: Proceedings of the 14th International Conference on Machine Learning, Nashville, TN, pp. 211–218 (1997)
17. Martínez-Muñoz, G., Suárez, A.: Pruning in ordered bagging ensembles. In: Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, pp. 609–616 (2006)

18. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
19. Tamon, C., Xiang, J.: On the boosting pruning problem. In: Proceedings of the 11th European Conference on Machine Learning, Barcelona, Spain, pp. 404–412 (2000)
20. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research 10, 271–289 (1999)
21. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (2000)
22. Zhang, Y., Burer, S., Street, W.N.: Ensemble pruning via semi-definite programming. Journal of Machine Learning Research 7, 1315–1338 (2006)
23. Zhou, Z.-H., Tang, W.: Selective ensemble of decision trees. LNCS (LNAI), vol. 2639, pp. 476–483. Springer, Heidelberg (2003)
24. Zhou, Z.-H., Wu, J., Tang, W.: Ensembling neural networks: Many could be better than all. Artificial Intelligence 137(1-2), 239–263 (2002)
25. Zhu, X.: Semi-supervised learning literature survey. Technical report, Department of Computer Sciences, University of Wisconsin Madison (2007)

Criteria Ensembles in Feature Selection

Petr Somol[1,2], Jiří Grim[1,2], and Pavel Pudil[2,1]

[1] Dept. of Pattern Recognition, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, 182 08 Prague, Czech Republic
{somol,grim}@utia.cas.cz
http://ro.utia.cas.cz/

[2] Faculty of Management, Prague University of Economics, Czech Republic
pudil@fm.vse.cz
http://www.fm.vse.cz

Abstract. In feature selection the effect of over-fitting may lead to serious degradation of generalization ability. We introduce the concept of combining multiple feature selection criteria in feature selection methods with the aim to obtain feature subsets that generalize better. The concept is applicable with many existing feature selection methods. Here we discuss in more detail the family of sequential search methods. The concept does not specify which criteria to combine – to illustrate its feasibility we give a simple example of combining the estimated accuracy of k-nearest neighbor classifiers for various k. We perform the experiments on a number of datasets. The potential to improve is clearly seen on improved classifier performance on independent test data as well as on improved feature selection stability.

1 Introduction

A common practice in multidimensional classification methods is to apply a feature selection (FS) procedure as the first preliminary step. The aim is to avoid overfitting in the training phase since, especially in the case of small and/or high-dimensional data, the classifiers tend to adapt to some specific properties of training data which are not typical for the independent test data. The resulting classifier then poorly generalizes and the classification accuracy on independent test data decreases [2]. By choosing a small subset of "informative" features we try to reduce the risk of overfitting and to improve the generalizing property of the classifier. Moreover, FS may also lead to data acquisition cost savings as well as to gains in processing speed.

In most cases a natural way to choose the optimal subset of features would be to minimize the probability of classification error. As the exact evaluation of error probability is usually not viable, we have to minimize some estimates of classification error (wrapper methods) or at least some estimates of its upper bound, or even some intuitive probabilistic criteria like entropy, model-based class distances, distribution divergences, etc. (filter methods) [7]. In order to avoid biased solutions the chosen criterion has to be evaluated on an independent validation set. Nevertheless, the problem of overfitting applies to FS criteria and FS algorithms as well [11] and cannot be fully avoided by means of validation. It is well known that different optimality criteria may choose different feature subsets [2]. The resulting feature subsets may differ even if one and the same criterion is applied to differently chosen training data. In this respect the "stability" of the resulting feature subsets becomes a relevant viewpoint [8] [14].

It has been shown repeatedly in literature that classification system performance may be considerably improved in some cases by means of classifier combination [6]. In multiple-classifier systems FS is often applied separately to yield different subsets for each classifier in the system [5] [4]. Another approach is to select one feature subset to be used in all co-operating classifiers [10] [3].

In contrary to such approaches we utilize the idea of combination to eventually produce one feature subset to be used with one classifier. We propose to combine FS criteria with the aim to obtain a feature subset that has better generalization properties than subsets obtained using single criteria. In the course of FS process we evaluate several criteria simultaneously and, at any selection step, the best features are identified by combining the criteria output. In the following we show that subsets obtained by combining selection criteria output using voting and weighted voting are more stable and improve the classifier performance on independent data in most cases.

1.1 Notation

Let Y denote the set of all $D = |\mathrm{Y}|$ features. Further let $X_d \subset \mathrm{Y}$ denote the current subset of d features, f_i denote the i-th feature in the set of all features, $i = 1, \ldots, D$ and $J(\cdot)$ denote a FS criterion. Without loss of generality we will assume that higher $J(\cdot)$ value indicates better feature subset.

2 Decomposing Sequential Search Methods

To simplify the discussion of the criterion combination scheme to be proposed let us focus only on the family of sequential search methods. Most of the known sequential FS algorithms share the same "core mechanism" of adding and removing features to/from a working subset. The respective algorithm steps can be described as follows (for the sake of simplicity we consider only non-generalized algorithms that process one feature at a time only):

Definition 1. *Let ADD() be the operation of adding feature f^+ to the working set X_d to obtain X_{d+1}:*

$$X_{d+1} = X_d \cup \{f^+\} = ADD(X_d), \qquad X_d, X_{d+1} \subset Y \tag{1}$$

where

$$f^+ = \arg \max_{f \in Y \setminus X_d} \mathcal{J}^+(X_d, f) \tag{2}$$

with $\mathcal{J}^+(X_d, f)$ denoting the criterion function used to evaluate the subset obtained by adding f, where $f \in Y \setminus X_d$, to X_d.

Definition 2. *Let REMOVE() be the operation of removing feature f^- from the working set X_d to obtain set X_{d-1}:*

$$X_{d-1} = X_d \setminus \{f^-\} = REMOVE(X_d), \qquad X_d, X_{d-1} \subset Y \tag{3}$$

where

$$f^- = \arg\max_{f \in X_d} \mathcal{J}^-(X_d, f) \tag{4}$$

with $\mathcal{J}^-(X_d, f)$ denoting the criterion function used to evaluate the subset obtained by removing f, where $f \in X_d$, from X_d.

In standard sequential FS methods the impact of feature adding (resp. removal) in one algorithm step is evaluated simply as follows:

$$\mathcal{J}^+(X_d, f) = J(X_d \cup \{f\}), \qquad \mathcal{J}^-(X_d, f) = J(X_d \setminus \{f\}) \ , \tag{5}$$

where $J(\cdot)$ is either a filter- or wrapper-based criterion [7] to be evaluated on the subspace defined by the tested feature subset.

2.1 Simplified View of Sequential Search Methods

In order to simplify the notation for a repeated application of FS operations we introduce the following useful notation

$$\begin{aligned} X_{d+2} &= ADD(X_{d+1}) = ADD(ADD(X_d)) = ADD^2(X_d) \ , \\ X_{d-2} &= REMOVE(REMOVE(X_d)) = REMOVE^2(X_d) \ , \end{aligned} \tag{6}$$

and more generally

$$X_{d+\delta} = ADD^\delta(X_d), \qquad X_{d-\delta} = REMOVE^\delta(X_d) \tag{7}$$

Using this notation we can now outline the basic idea behind sequential FS algorithms very simply. For instance:

SFS (*Sequential Forward Selection* [16] yielding a subset of t features):

1. $X_t = ADD^t(\emptyset)$.

SFFS (*Sequential Forward Floating Selection* [9] yielding a subset of t features, with optional search-restricting parameter $\Delta \in [0, D-t]$):

1. Start with $X_0 = \emptyset$, $d = 0$.
2. $X_{d+1} = ADD(X_d)$, $d = d + 1$.
3. Repeat $X_{d-1} = REMOVE(X_d)$, $d = d - 1$ as long as it improves solutions already known for the lower d.
4. If $d < t + \Delta$ go to 2.

OS (*Oscillating Search* [13] yielding a subset of t features, with optional search-restricting parameter $\Delta \geq 1$):

1. Start with initial set X_t of t features. Set cycle depth to $\delta = 1$.
2. Let $X_t^{\downarrow} = ADD^{\delta}(REMOVE^{\delta}(X_t))$.
3. If $X_t^{\downarrow}$ better than X_t, let $X_t = X_t^{\downarrow}$, let $\delta = 1$ and go to 2.
4. Let $X_t^{\uparrow} = REMOVE^{\delta}(ADD^{\delta}(X_t))$.
5. If $X_t^{\uparrow}$ better than X_t, let $X_t = X_t^{\uparrow}$, let $\delta = 1$ and go to 2.
6. If $\delta < \Delta$ let $\delta = \delta + 1$ and go to 2.

DOS (*Dynamic Oscillating Search* [15] yielding a subset of optimized size p, with optional search-restricting parameter $\Delta \geq 1$):

1. Start with $X_p = ADD(ADD(\emptyset))$, p=2. Set cycle depth to $\delta = 1$.
2. Compute $ADD^{\delta}(REMOVE^{\delta}(X_t))$; if any intermediate subset X_i, $i \in [p-\delta, p]$ is found better than X_p, let it become the new X_p with $p = i$, let $\delta = 1$ and restart step 2.
3. Compute $REMOVE^{\delta}(ADD^{\delta}(X_t))$; if any intermediate subset X_j, $j \in [p, p+\delta]$ is found better than X_p, let it become the new X_p with $p = j$, let $\delta = 1$ and go to 2.
4. If $\delta < \Delta$ let $\delta = \delta + 1$ and go to 2.

Obviously, other FS methods can be described using the notation above as well.

3 Combining Multiple Criteria

Different criterion functions may reflect different properties of the evaluated feature subsets. Incorrectly chosen criterion may easily lead to the wrong subset. Combining multiple criteria is justifiable from the same reasons as traditional multiple classifier systems. It should reduce the tendency to over-fit by preferring features that perform well with respect to several various criteria instead of just one and consequently enable to improve the generalization properties of the selected subset of features. The idea is to reduce the possibility of a single criterion to exploit too strongly the specific properties of training data, that may not be present in independent test data.

In the following we discuss several straight-forward approaches to criteria combination by means of re-defining $\mathcal{J}^{+}$ and $\mathcal{J}^{-}$ in Definitions 1 and 2. We will consider ensembles of arbitrary feature selection criteria $J^{(k)}$, $k = 1, \ldots, K$. In Section 4 concrete examples will be given for $J^{(k)}$, $k = 1, \ldots, 4$ standing for the accuracy of $(2k-1)$-Nearest Neighbor classifier.

3.1 Simplest Criterion Combination

First let us discuss the simplest combination option. To realize a simple criterion ensemble consisting of criteria $J^{(k)}$, $k = 1, \ldots, K$, consider modifying Definitions 1 and 2 as follows

$$\mathcal{J}^{+}_{\text{avg}}(X_d, f) = \frac{1}{K} \sum_{k=1}^{K} J^{(k)}(X_d \cup \{f\}) \tag{8}$$

$$\mathcal{J}^{-}_{\text{avg}}(X_d, f) = \frac{1}{K} \sum_{k=1}^{K} J^{(k)}(X_d \setminus \{f\}) ,$$

or, to put more preference on features that generally "fail the least" with respect to all of the considered criteria, modify Definitions 1 and 2 as follows

$$\mathcal{J}^{+}_{\min}(X_d, f) = \min_{k=1,\ldots,K} J^{(k)}(X_d \cup \{f\}) \tag{9}$$
$$\mathcal{J}^{-}_{\min}(X_d, f) = \min_{k=1,\ldots,K} J^{(k)}(X_d \setminus \{f\}) \ .$$

Remark: Maximizing would meaninglessly emphasize feature over-selection.

Clearly, none of the approaches (8) and (9) is applicable unless all $J^{(k)}$, $k = 1, \ldots, K$ yield equally bounded values. This should apparently be no problem with wrappers, where the estimated classification accuracy can be easily normalized to $[0, 1]$. However, both (8) and (9) produce feature preferences that are hard to interpret, especially if the used criteria $J^{(k)}$, $k = 1, \ldots, K$ tend to yield values of differing size (albeit equally bounded). Accordingly, no consistent advantage over single-criterion FS has been observed throughout the numerous experiments we have performed. Therefore, the simple criterion value combination as described in this Section *is to be considered unsatisfactory* and unable to bring reliable improvement over the traditional single-criterion FS methods.

3.2 Multiple Criterion Voting

A better way to realize the idea of criterion ensemble is to implement a form of voting. The intention is to reveal stability in feature preferences, with no restriction on the principle or behavior of the combined criteria $J^{(k)}$, $k = 1, \ldots, K$. Accordingly, we will redefine $\mathcal{J}^{+}$ and $\mathcal{J}^{-}$ to express averaged feature ordering preferences instead of directly combining criterion values.

In the following we define $\mathcal{J}^{+}_{order}$ as replacement of $\mathcal{J}^{+}$ in Definition 1. The following steps are to be taken separately for each criterion $J^{(k)}$, $k = 1, \ldots, K$ in the considered ensemble of criteria. First, evaluate all values $J^{(k)}(X_d \cup \{f_i\})$ for $i = 1, \ldots, D-d$, where $f_i \in Y \setminus X_d$. Next, order these values descending with possible ties resolved arbitrarily at this stage and encode the ordering using indexes $i_j, j = 1, \ldots, D-d, i_j \in [1, D-d]$ where $i_m \neq i_n$ for $m \neq n$:

$$J^{(k)}(X_d \cup \{f_{i_1}\}) \geq J^{(k)}(X_d \cup \{f_{i_2}\}) \geq \cdots \geq J^{(k)}(X_d \cup \{f_{i_{D-d}}\}) \ . \tag{10}$$

Next, express feature preferences using coefficient $\alpha^{(k)}_j$, $j = 1, \ldots, D-d$, defined to take into account possible feature preference ties as follows:

$$\alpha^{(k)}_{i_1} = 1 \tag{11}$$
$$\alpha^{(k)}_{i_j} = \begin{cases} \alpha^{(k)}_{i_{j-1}} & \text{if } J^{(k)}(X_d \cup \{f_{i_{(j-1)}}\}) = J^{(k)}(X_d \cup \{f_{i_j}\}) \\ \alpha^{(k)}_{i_{j-1}} + 1 & \text{if } J^{(k)}(X_d \cup \{f_{i_{(j-1)}}\}) > J^{(k)}(X_d \cup \{f_{i_j}\}) \end{cases} \quad \text{for } \ j \geq 2 \ .$$

Now, having collected the values $\alpha^{(k)}_j$ for all $k = 1, \ldots, K$ and $j = 1, \ldots, D-d$ we can transform the criteria votes to a form usable in Definition 1 by defining:

$$\mathcal{J}^{+}_{order}(X_d, f_i) = -\frac{1}{K}\sum_{k=1}^{K} \alpha^{(k)}_i \ . \tag{12}$$

The definition of $\mathcal{J}^{-}_{order}$ is analogous.

3.3 Multiple Criterion Weighted Voting

Suppose we introduce an additional restriction to the values yielded by criteria $J^{(k)}$, $k = 1, \ldots, K$ in the considered ensemble. Suppose each $J^{(k)}$ yields values from the same interval. This is easily fulfilled, e.g., in wrapper methods where the estimated correct classification rate is usually normalized to $[0, 1]$. Now the differences between $J^{(k)}$ values (for fixed k) can be treated as weights expressing relative feature preferences of criterion k. In the following we define $\mathcal{J}^{+}_{weigh}$ as replacement of $\mathcal{J}^{+}$ in Def. 1. The following steps are to be taken separately for each criterion $J^{(k)}$, $k = 1, \ldots, K$ in the considered ensemble of criteria. First, evaluate all values $J^{(k)}(X_d \cup \{f_i\})$ for fixed k and $i = 1, \ldots, D - d$, where $f_i \in Y \setminus X_d$. Next, order the values descending with possible ties resolved arbitrarily at this stage and encode the ordering using indexes $i_j, j = 1, \ldots, D - d$ in the same way as shown in (10). Now, express feature preferences using coefficient $\beta^{(k)}_j$, $j = 1, \ldots, D - d$ defined to take into account the differences between the impact the various features from $Y \setminus X_d$ have on the criterion value:

$$\beta^{(k)}_{i_j} = J^{(k)}(X_d \cup \{f_{i_1}\}) - J^{(k)}(X_d \cup \{f_{i_j}\}) \text{ for } j = 1, \ldots, D - d \; . \qquad (13)$$

Now, having collected the values $\beta^{(k)}_j$ for all $k = 1, \ldots, K$ and $j = 1, \ldots, D-d$ we can transform the criteria votes to a form usable in Definition 1 by defining:

$$\mathcal{J}^{+}_{weigh}(X_d, f_i) = -\frac{1}{K} \sum_{k=1}^{K} \beta^{(k)}_i \; . \qquad (14)$$

The definition of $\mathcal{J}^{-}_{weigh}$ is analogous.

3.4 Resolving Voting Ties

Especially in small sample data where the discussed techniques are of particular importance it may easily happen that

$$\mathcal{J}^{+}_{order}(X_d, f_i) = \mathcal{J}^{+}_{order}(X_d, f_j) \quad \text{for} \quad i \neq j \; . \qquad (15)$$

(The same can happen for $\mathcal{J}^{-}_{order}$, $\mathcal{J}^{+}_{weigh}$, $\mathcal{J}^{-}_{weigh}$.) To resolve such ties we employ an additional mechanism. To resolve $\mathcal{J}^{+}$ ties we collect in the course of FS process for each feature f_i, $i = 1, \ldots, D$ the information about all values (12) evaluated so far. In case of $\mathcal{J}^{+}$ ties the feature with higher average over previous values (12) is preferred. (Tie resolution for J^{-}_{order}, J^{+}_{weigh}, J^{-}_{weigh} is analogous.)

4 Experimental Results

We performed a series of FS experiments on various data-sets from UCI repository [1] and one data-set (xpxinsar satellite) from Salzburg University. Many of the data-sets have small sample size with respect to dimensionality. In this type of problems any improvement of generalization properties plays crucial role. To put the robustness of the proposed criterion voting schemes on test we used in all experiments the *Dynamic Oscillating Search* algorithm [15] as one of the strongest available subset optimizers, with high risk of over-fitting.

To illustrate the concept we have resorted in all experiments to combining classification accuracy of four simple wrappers – *k-Nearest Neighbor* (k-NN) classifiers for $k = 1, 3, 5, 7$, as the effects of increasing k are well understandable. With increasing k the k-NN class-separating hyperplane gets smoother – less affected by outliers but also less sensitive to possibly important detail.

Each experiment was run using 2-tier cross-validation. In the "outer" 10-fold cross-validation the data was repeatedly split to 90% training part and 10%

Table 1. ORDER VOTING. Comparing single-criterion and multiple-criterion FS (first and second row for each data-set). All reported classification rates obtained using 3-NN classifier on independent test data. Improvement emphasized in bold (the higher the classification rate and/or stability measures' value the better).

Data	Dim.	Classes	Rel. sample size	FS Wrapper(s)	Classsif. rate		Subset size d		FS Stability				FS time
					Mean	*S.Dv.*	Mean	*S.Dv.*	C	CW	CW $_{rel}$	ATI (GK)	h:m:s
derm	36	6	1.657	3-NN	.970	*.023*	9.6	*0.917*	.481	.664	.597	.510	00:03:24
				1,3,5,7-NN	**.978**	*.027*	10.7	*1.676*	.406	.636	.534	.486	00:15:50
hous	14	5	7.229	3-NN	.707	*.088*	4.9	*1.513*	.308	.617	.456	.478	00:01:19
				1,3,5,7-NN	.689	*.101*	5.4	*1.744*	**.389**	**.650**	**.497**	**.509**	00:04:48
iono	34	2	5.162	3-NN	.871	*.078*	5.6	*1.500*	.200	.349	.303	.216	00:02:10
				1,3,5,7-NN	**.882**	*.066*	4.7	*1.269*	**.262**	**.454**	**.441**	**.325**	00:06:09
mammo	65	2	0.662	3-NN	.821	*.124*	4.2	*1.833*	.248	.476	.497	.343	00:00:30
				1,3,5,7-NN	**.846**	*.153*	3	*1.483*	**.306**	**.519**	**.519**	**.420**	00:01:23
opt38	64	2	8.773	3-NN	.987	*.012*	9	*1.414*	.192	.449	.412	.297	01:34:14
				1,3,5,7-NN	.987	*.012*	9.5	*1.360*	**.219**	**.512**	**.490**	**.362**	06:22:00
sati	36	6	20.532	3-NN	.854	*.031*	14.2	*3.156*	.367	.557	.347	.392	32:59:47
				1,3,5,7-NN	**.856**	*.037*	14.5	*3.801*	**.392**	**.567**	**.357**	**.399**	116:26:
segm	19	7	17.368	3-NN	.953	*.026*	4.7	*1.735*	.324	.648	.610	.550	00:35:13
				1,3,5,7-NN	**.959**	*.019*	4.6	*2.245*	.282	**.652**	**.625**	**.601**	02:02:40
sonar	60	2	1.733	3-NN	.651	*.173*	12.8	*4.895*	.244	.411	.327	.260	00:07:15
				1,3,5,7-NN	**.676**	*.130*	8.8	*4.020*	.185	.389	**.350**	.260	00:16:02
specf	44	2	3.034	3-NN	.719	*.081*	9.5	*4.522*	.160	.281	.174	.157	00:03:56
				1,3,5,7-NN	**.780**	*.111*	9.8	*3.092*	**.210**	**.358**	**.255**	**.237**	00:15:36
wave	40	3	41.667	3-NN	.814	*.014*	17.2	*2.561*	.486	.792	.680	.657	62:36:30
				1,3,5,7-NN	**.817**	*.011*	16.4	*1.356*	.477	**.826**	**.753**	**.709**	70:27:36
wdbc	30	2	9.483	3-NN	.965	*.023*	10.3	*1.676*	.329	.507	.327	.345	00:12:18
				1,3,5,7-NN	**.967**	*.020*	10.1	*3.176*	**.338**	**.530**	**.360**	**.375**	00:41:07
wine	13	3	4.564	3-NN	.966	*.039*	5.9	*0.831*	.544	.731	.568	.594	00:00:15
				1,3,5,7-NN	.960	*.037*	6	*1.000*	**.556**	**.748**	**.575**	**.606**	00:00:54
wpbc	31	2	3.194	3-NN	.727	*.068*	9.1	*3.048*	.226	.347	.168	.211	00:01:53
				1,3,5,7-NN	.727	*.056*	7.2	*2.600*	.197	.312	**.189**	.188	00:04:41
xpxi	57	7	4.313	3-NN	.895	*.067*	10.8	*1.939*	.434	.648	.618	.489	05:07:06
				1,3,5,7-NN	.894	*.069*	11.5	*3.233*	.421	**.657**	**.630**	**.495**	21:19:43

Table 2. WEIGHTED VOTING. Comparing single-criterion and multiple-criterion FS (first and second row for each data-set). All reported classification rates obtained using 3-NN classifier on independent test data. Improvement emphasized in bold (the higher the classification rate and/or stability measures' value the better).

Data	Dim.	Classes	Rel. sample size	FS Wrapper(s)	Classsif. rate		Subset size d		FS Stability				FS time
					Mean	S.Dv.	Mean	S.Dv.	C	CW	CW_{rel}	ATI (GK)	h:m:s
derm	36	6	1.657	3-NN	.970	*.023*	9.6	*0.917*	.481	.664	.597	.510	00:03:24
				1,3,5,7-NN	**.978**	*.017*	10.3	*1.552*	**.491**	**.721**	**.658**	**.573**	00:17:42
hous	14	5	7.229	3-NN	.707	*.088*	4.9	*1.513*	.308	.617	.456	.478	00:01:19
				1,3,5,7-NN	**.716**	*.099*	5.6	*2.29*	**.455**	**.639**	**.459**	**.495**	00:03:33
iono	34	2	5.162	3-NN	.871	*.078*	5.6	*1.500*	.200	.349	.303	.216	00:02:10
				1,3,5,7-NN	**.897**	*.059*	4.9	*1.758*	**.278**	**.426**	**.393**	**.345**	00:07:40
mammo	65	2	0.662	3-NN	.821	*.124*	4.2	*1.833*	.248	.476	.497	.343	00:00:30
				1,3,5,7-NN	.813	*.153*	2.6.	*1.428*	.210	**.487**	**.542**	**.390**	00:00:43
opt38	64	2	8.773	3-NN	.987	*.012*	9	*1.414*	.192	.449	.412	.297	01:34:14
				1,3,5,7-NN	**.988**	*.011*	8.6	*1.020*	**.304**	**.576**	**.569**	**.423**	07:39:33
sati	36	6	20.532	3-NN	.854	*.031*	14.2	*3.156*	.367	.557	.347	.392	32:59:47
				1,3,5,7-NN	**.856**	*.038*	13.8	*2.182*	**.400**	**.618**	**.448**	**.456**	99:30:44
segm	19	7	17.368	3-NN	.953	*.026*	4.7	*1.735*	.324	.648	.610	.550	00:35:13
				1,3,5,7-NN	**.959**	*.019*	4.6	*2.245*	**.354**	**.667**	**.644**	**.610**	02:26:29
sonar	60	2	1.733	3-NN	.651	*.173*	12.8	*4.895*	.244	.411	.327	.260	00:07:15
				1,3,5,7-NN	.614	*.131*	10.1	*3.015*	.192	.361	.301	.224	00:20:32
specf	44	2	3.034	3-NN	.719	*.081*	9.5	*4.522*	.160	.281	.174	.157	00:03:56
				1,3,5,7-NN	**.787**	*.121*	9.1	*3.590*	**.205**	**.369**	**.285**	**.229**	00:17:54
wave	40	3	41.667	3-NN	.814	*.014*	17.2	*2.561*	.486	.792	.680	.657	62:36:30
				1,3,5,7-NN	.814	*.016*	16.9	*1.700*	**.560**	**.822**	**.727**	**.700**	287:06:
wdbc	30	2	9.483	3-NN	.965	*.023*	10.3	*1.676*	.329	.507	.327	.345	00:12:18
				1,3,5,7-NN	**.967**	*.020*	10.3	*4.267*	**.347**	**.524**	**.352**	**.346**	00:55:08
wine	13	3	4.564	3-NN	.966	*.039*	5.9	*0.831*	.544	.731	.568	.594	00:00:15
				1,3,5,7-NN	.960	*.037*	6.6	*1.200*	**.606**	**.741**	.567	**.606**	00:00:28
wpbc	31	2	3.194	3-NN	.727	*.068*	9.1	*3.048*	.226	.347	.168	.211	00:01:53
				1,3,5,7-NN	.686	*.126*	6.9	*2.508*	.196	.322	**.211**	.192	00:04:24
xpxi	57	7	4.313	3-NN	.895	*.067*	10.8	*1.939*	.434	.648	.618	.489	05:07:06
				1,3,5,7-NN	.895	*.071*	11	*2.683*	**.444**	.638	.595	.475	38:35:53

testing part. FS was done on the training part. Because we used *wrapper* setup, each criterion evaluation involved training and testing classifier(s). To utilize the training data better, it was processed by means of "inner" 10-fold cross-validation, i.e., was repeatedly split to 90% part used for classifier training and 10% part used for classifier validation. The averaged classifier accuracy then served as single FS criterion output. Each selected feature subset was eventually evaluated on the 3-NN classifier, trained on the training part and tested on the testing part of the "outer" data split. The resulting classification accuracy, averaged over "outer" data splits, is reported in Tables 1 and 2.

In both Tables 1 and 2 for each data-set the multiple-criterion results (second row) are compared to the single-criterion result (first row) obtained using 3-NN as wrapper. For each data-set its basic parameters are reported, including its class-averaged dimensionality-to-class-size ratio. Note that in each of the "outer" runs possibly different feature subset can be selected. The stability of feature preferences across the "outer" cross-validation runs has been evaluated using the stability measures C, CW, CW_{rel} and ATI (a.k.a. GK), all yielding values from $[0, 1]$, which reflect various aspects of the stability problem as described in [14]. We also report the total time needed to complete each 2-tier cross-validation single-threaded experiment on an up-to-date AMD Opteron CPU.

Table 1 illustrates the impact of multiple criterion voting (12) as described in Section 3.2. Table 2 illustrates the impact of multiple criterion weighted voting (14) as described in Section 3.3. Improvement is emphasized in bold. The results presented in both Tables 1 and 2 clearly show that the concept of criteria ensemble has the potential to improve both the generalization ability (as illustrated by improved classification accuracy on independent test data) and FS stability (sensitivity to perturbations in training data). The positive effect of either (12) or (14) is not present in all cases (in some cases the performance degraded) but it is clearly prevalent among the tested datasets.

It can be also seen that none of the presented schemes can be identified as the better choice. Moreover, care should be taken when applying either of the two, as the criterion ensemble effect may be even counterproductive as was the case of *house* dataset in Table 1 and *sonar* and *wpbc* datasets in Table 2.

5 Concluding Remarks

It has been shown that combining multiple critera by voting in FS process has the potential to improve both the generalization properties of the selected feature subsets as well as the stability of feature preferences. The actual gain is problem dependent and can not be guaranteed, although the improvement on some datasets is substantial.

The idea of combining FS criteria by voting can be applied not only in sequential selection methods but generally in any FS method where a choice is made among several candidate subsets (generated, e.g., randomly as in genetic algorithms). Additional means of improving robustness can be considered, e.g., ignoring the best and worst result among all criteria, etc.

Acknowledgements. The work has been supported by GAČR grant 102/08/0593 and CR MŠMT grants 2C06019 ZIMOLEZ and 1M0572 DAR.

References

1. Asuncion, A., Newman, D.: UCI machine learning repository (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html
2. Devijver, P.A., Kittler, J.: Pattern Recognition: A Statistical Approach. Prentice-Hall International, London (1982)
3. Dutta, D., Guha, R., Wild, D., Chen, T.: Ensemble Feature Selection: Consistent Descriptor Subsets for Multiple QSAR Models. J. Chem. Inf. Model. 47(3), 989–997 (2007)

4. Emmanouilidis, C., Hunter, A., MacIntyre, J., Cox, C.: Multiple-criteria genetic algorithms for feature selection inneuro-fuzzy modeling. In: Proc. Int. Joint Conf. on Neural Networks, vol. (6), pp. 4387–4392 (1999)
5. Günter, S., Bunke, H.: Feature selection algorithms for the generation of multiple classifier systems and their application to handwritten word recognition. Pattern Recogn. Lett. 25(11), 1323–1336 (2004)
6. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On Combining Classifiers. IEEE Trans. Pattern Anal. Mach. Intell. 20(3), 226–239 (1998)
7. Kohavi, R., John, G.: Wrappers for feature subset selection. Artificial Intelligence 97, 273–324 (1997)
8. Kuncheva, L.I.: A stability index for feature selection. In: Proc. 25th IASTED Int. Multi-Conf. Artificial Intelligence and Applications, pp. 421–427 (2007)
9. Pudil, P., Novovičová, J., Kittler, J.: Floating search methods in feature selection. Pattern Recognition Letters 15, 1119–1125 (1994)
10. Raykar, V.C., Krishnapuram, B., Bi, J., Dundar, M., Rao, R.B.: Bayesian multiple instance learning: automatic feature selection and inductive transfer. In: Proc. 25th Int. Conf. on Machine Learning, pp. 808–815 (2008)
11. Raudys, S.: Feature over-selection. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR 2006 and SPR 2006. LNCS, vol. 4109, pp. 622–631. Springer, Heidelberg (2006)
12. Saeys, Y., Abeel, T., de Peer, Y.V.: Towards robust feature selection techniques. In: Proceedings of Benelearn, pp. 45–46 (2008)
13. Somol, P., Pudil, P.: Oscillating search algorithms for featute selection. In: Proc. 15th IAPR Int. Conference on Pattern Recognition, pp. 406–409 (2000)
14. Somol, P., Novovičová, J.: Evaluating the stability of feature selectors that optimize feature subset cardinality. In: Proc. SSPR/SPR. LNCS, vol. 5342, pp. 956–966. Springer, Heidelberg (2008)
15. Somol, P., Novovičová, J., Pudil, P., Grim, J.: Dynamic oscillating search algorithm for feature selection. In: Proc. 19th IAPR Int. Conf. on Pattern Recognition. IEEE Computer Society Press, Tampa (2008) file: WeAT9.15.pdf
16. Whitney, A.W.: A direct method of nonparametric measurement selection. IEEE Trans. Comput. 20(9), 1100–1103 (1971)

Network Protocol Verification by a Classifier Selection Ensemble

Francesco Gargiulo[1], Ludmila I. Kuncheva[2], and Carlo Sansone[1]

[1] Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli Federico II, Via Claudio, 21 I-80125 Napoli, Italy
{francesco.grg,carlosan}@unina.it
[2] School of Computer Science, University of Bangor, UK
l.i.kuncheva@bangor.ac.uk

Abstract. Classical approaches for network traffic classification are based on port analysis and packet inspection. Recent studies indicate that network protocols can be recognised more accurately using the flow statistics of the TCP connection. We propose a classifier selection ensemble for a fast and accurate verification of network protocols. Using the requested port number, the classifier selector directs the decision to an ensemble member responsible for this port. The chosen ensemble member ramifies the decision further using the "sign pattern" of the first four packets. Finally, a decision tree classifier labels the flow as 'accepted' or 'rejected' using the sizes of the first four packets. The ensemble has modular architecture which allows further modules to be individually trained and added. The classifiers were cross-tested using designated training and testing data of network traffic traces from three institutions. The results show that accuracy need not be sacrificed for speed of classification, and that the protocol classification is robust from one network to another.

1 Introduction

Network traffic classification is important for ensuring quality of service (QoS), security, optimal priority assignment, and general traffic management. Fast and accurate catching of an inadequate application protocol is imperative when security is concerned. Ideally, a smart firewall would block such protocols at their onset. The basic traffic unit we consider in this paper is termed a **flow**. We define a flow as a bi-directional ordered sequence of packets with the same IP addresses and TCP port numbers. A flow is either accepted by the classifier as one of the valid protocols or rejected as unknown or known but non-allowed. Usually, to label a flow, the information of all packets is needed. For this information to be extracted, the flow must have entered the network, so the classification would come when it may be too late to decline service.

The main approaches to traffic classification are port-based, payload-based and statistical.

The **port-based** approach uses the assumption that each port is associated with one protocol [13]. The protocol classification is made simply by reading the port number. However, this method is not suitable for networks with dynamic port allocation. For such networks an undesirable *application* may be directed through a port conventionally associated with an acceptable application. Thus the classification of applications cannot be done from the port number only.

Passing a non-allowed protocol through an accepted port may also be a result of malicious activity. Another problem with the port-based approach is that it will not prevent 'tunnelling', i.e., protocol X embedded within and disguised as protocol Y, where Y an accepted protocol for the network while X might not be [3,5].

The **payload-based** approach looks at the content of the packets [15]. The protocol verification is more accurate but requires more computational resources. An adverse issue associated with the payload-based approach is related to the privacy of the content. Also, when the traffic is encrypted the approach will not work [9].

The **statistical** approach takes characteristics of the flow as the input features, e.g., number of packets; their length; minimum, maximum and average length, etc. Various statistical classifiers have been tried on the extracted features, e.g., Bayesian networks [1], Support Vector Machines [12], Gaussian Mixture Modelling and Decision Trees [6]. Statistical tests such as goodness of fit, χ^2 and Kolmogorov-Smirnov have also been tried [7] to single out anomalies such as incidents of `skype` application within an `http` protocol. Extracting discriminative features is a major focus of the works on statistical traffic classification [1,12]. It is worth noting that most of the statistical approaches proposed so far [1,6,16] need to be retrained when the number of allowed protocols varies.

The main problem of both the payload-based and the statistical approaches is that traffic flows can only be classified once they have passed through the system completely. This limits their applicability for online classification. Anyway, it has been recently shown that accurate classification can be achieved using only the sizes and the directions of the first few packets of the TCP connection [2,6].

The requirement for operational speed brings in the idea of classifier *selection* ensemble where only one of a set of 'experts' has to make a decision [11]. The ensemble consists of member classifiers (experts) and an 'oracle' that authorises one of the classifiers to pass its decision as the ensemble decision. The oracle may have pre-defined regions of competence for the classifiers [14] or dynamically allocated regions [17]. With dynamic competence allocation the suggested labels for the object of interest $\mathbf{x}$ are further analysed using past data. The classifier whose predicted label has been most accurate for the neighbourhood of $\mathbf{x}$ is chosen to produce the ensemble decision. While dynamic allocation has been found to be very successful, it requires that all ensemble members classify $\mathbf{x}$. Besides, past data needs to be stored and searched through. Since we are aiming at a fast classification, we propose to use pre-defined competence regions and train a bespoke classifier for each region. We propose to use the port number (pretend protocol name) as the oracle determining the regions of competence. The directions and sizes of the first four packets of the TCP flow are then used as the features in a further 2-stage classifier. The features and the modular architecture were chosen so that the classification is both fast and accurate, and new modules can be trained and added to the system without re-training any already trained part.

The rest of the paper is organised as follows. Section 2 describes the proposed system and Section 3 shows the experimental results. Our conclusions and future plans are given in Section 4.

2 A Classifier Selection Ensemble for Network Traffic Classification

We propose the following classifier selection ensemble. Each port number has a classifier trained to verify that the traffic through that port follows the expected protocol. Thus the classifier selector only checks the port number and directs the flow to the respective classifier. If the port number is not one of the pre-defined set the flow is rejected.

2.1 The Features

Following [1],[6] and [7], we propose to use only the first four packets and record the following features:

- x_0, the pretend name of the flow guessed from the port number;
- x_1, x_2, x_3, x_4, the directions of the first four packets, $x_i \in \{0, 1\}$, where 0 means that the packet is transferred from server to client, and 1, from client to server;
- s_1, s_2, s_3, s_4, the payload sizes of the first four packets, where s_i are positive integers. As in [6], we leave off packets without payload because they are mostly used to exchange connection state information.

The generic architecture of the classifier ensemble is shown in Figure 1.

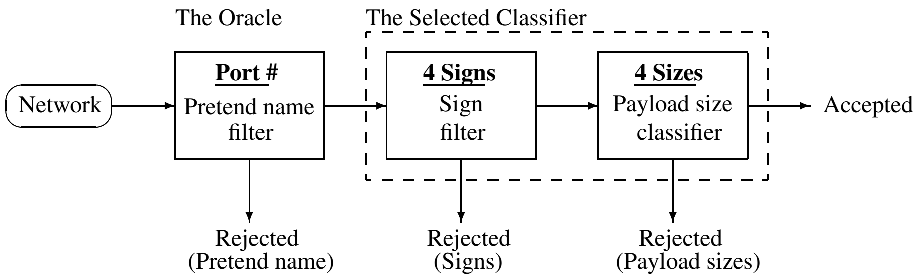


Fig. 1. The generic classifier ensemble architecture. Only the selected ensemble member is shown. Each ensemble member is implemented as a cascade classifier with 2 stages.

2.2 Classifier Selector: The Pretend Name

The port information is often neglected in classifying network traffic flow [6]. In this study we use this information in two ways. First, we label as unknown all flows whose port number (pretend name) does not appear in a pre-set list. Second, the pretend name is used to branch out the classification to a bespoke classifier. This partitions the feature space on the value of x_0, thereby reducing a multi-class problem to a two-class problem: match versus mismatch of the pretend name.

2.3 Ensemble Classifier – Stage 1: Sign Filter

To illustrate the consecutive steps of the system design we use a data set consisting of network traffic traces at the University of Brescia, Italy [6]. The data set is divided into training (58 478 instances) and testing (75 163 instances). The known protocols in the training data are: `pop3`, `smtp`, `http`, `msn`, `ftp` and `BitTorrent`. The testing data contains an additional class named 'unknown'. Table 1 shows a summary of the training data.

Table 1. Summary of the network traffic data (training) from the University of Brescia, Italy

Signs	Protocol and port number					
	`pop3`	`ftp`	`smtp`	`msn`	`BitTorr`	`http`
1 2 3 4	110	21	25	1863	6881	80
0 0 0 0	0	138	16	0	0	3
0 0 0 1	1	75	55	0	0	0
0 0 1 0	21	216	543	0	0	0
0 0 1 1	0	0	4	0	1	0
0 1 0 0	749	21	604	1	0	0
0 1 0 1	18823	5845	18186	0	1	0
0 1 1 0	17	1	18	0	1	0
0 1 1 1	0	0	1	0	0	0
1 0 0 0	0	0	0	328	23	5348
1 0 0 1	0	0	0	30	520	240
1 0 1 0	0	0	0	660	3609	826
1 0 1 1	0	0	0	4	753	12
1 1 0 0	0	0	0	1	8	427
1 1 0 1	0	0	0	0	87	76
1 1 1 0	0	0	0	0	9	108
1 1 1 1	0	0	0	0	45	23

The table shows that groups of protocols can be distinguished by the signs of the first four packets. For example, protocols `msn` (1863), `BitTorrent` (6881) and `http` (80) hardly ever begin with a packet from sever to client ($x_1 = 0$). The 7 exceptions in the table (out of 13 144 flows) may be thought of as recording mistakes. The distribution of the data suggests that the four signs can be used to filter out very quickly protocols that clearly do not match their pretend name. This constitutes the second stage of the cascade classifier, called the sign filter. A rejection threshold p_r is chosen next. The occurrences if each protocol (a column in Table 1) are scaled to form a probability distribution across the 16 sign combinations. All values with likelihood less than the chosen threshold are treated as outliers. Thus for each protocol, there are "impossible" sign combinations which make up the filter for that pretend name. For example, with threshold $p_r = 0.02$, the "allowed" combination of signs for the `http` protocol (80) are 1000, 1001, 1010, and 1100. All other protocols will be rejected by the sign filter.

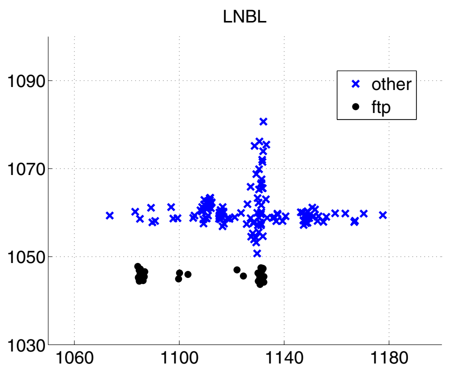


Fig. 2. Scatterplot of the LBNL training data with sign pattern 0100 in the plane spanned by the first two size features, s_1 and s_2

2.4 Ensemble Classifier – Stage 2: Decision Tree Classifier Using Payload Sizes

A separate classifier is trained for each sign combination that passes through the sign filter. For example, consider a protocol with pretend name `pop3` (110) and sign pattern 0100. Figure 2 shows a scatterplot of the data of network traffic traces from the Lawrence Berkeley National Laboratory (LBNL) with sign pattern 0100. The data points from `pop3` form a distinctive oblong cluster, away from the '+'–shaped cluster of the other protocols with the same sign pattern.[1]

Figure 3, on the other hand, displays the same scatterplot for the training data from the University of Brescia (UNIBS) and the Cooperative Association for Internet Data Analysis (CAIDA). The `ftp` protocol is also shown because it is present in these two data sets.

The figures show that:

1. Protocols `pop3` and `ftp` are very close to one another. To build a good classifier, both protocols should be present in the training data.
2. The classes have intricate irregular shapes (UNIBS and CAIDA data) which suggests that a decision tree classifier may fare well for this problem.
3. The class 'other' is different from one data set to another. The geometrical configuration of this class will depend on what protocols are accepted in the network. Note that class `ftp` in Figure 3 is, in fact, part of class 'other'.
4. Curiously, even the same class (`pop3`) has different appearances for the three different networks. This means that an ensemble has to be trained individually for each network. Hence an ensemble trained on the UNIBS data cannot be expected to be overly accurate on LBNL and CAIDA data, and vice versa.

Our ensemble differs from the stereotype in that a flow can be classified as "unaccepted" at each stage: the combiner (classifier selector), the sign filter and the decision tree

[1] Since the payload size is a discrete variable, multiple points may share both coordinates (x, y). A small random noise is added to all data so that the points move slightly off (x, y) in a random way. This will create an impression of the density of the data.

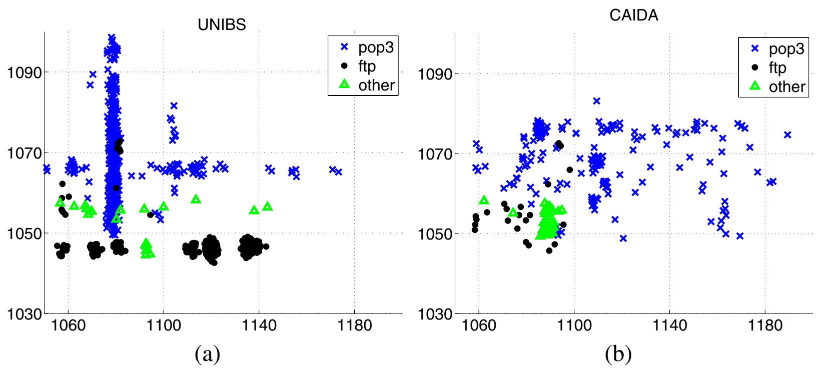


Fig. 3. Scatterplot of the UNIBS and CAIDA training data with sign pattern 0100 in the plane spanned by the first two size features, s_1 and s_2

classifier. This speeds up the decision process, which is important for online traffic classification.

3 Experimental Evaluation

A summary of the content of the three data sets used in this study is given in Table 2.

For the experimental evaluation we chose the three protocols that are common to all three data sets, `http`, `smtp` and `pop3`. A pilot experimental study on the UNIBS training data, using Weka [8], reinforced our choice of the decision tree classifier. Further to that, we carried out the following sets of experiments, where A, B, C refer to the three protocols and a, b, c refer to the three data sets:

(1). Train a classifier for a protocol with pretend name A using training data set a. To do this, assume that all flows have pretend name A so as to form a training data set with class labels 'A' and 'other'. Identify the sign patterns relevant for protocol A. Filter the data for each sign pattern. Using this data, train a decision tree classifier to distinguish between the two classes.
(2). Test the classifier on testing data sets b and c.
(3). Repeat steps (1) and (2) for protocols B and C

In this imaginary scenario, all traffic takes the pretend identity of the protocol in question. In reality, the likelihood of class 'other' will be much smaller. Therefore the classification accuracy achieved in the experiments is a pessimistic estimate of the accuracy expected during operation. Because of the specific experimental set-up, a direct comparison with classification accuracies obtained elsewhere may be misleading.

The cross-data classification accuracies are shown in Table 3.

Figure 4 gives the plot of Sensitivity versus 1−Specificity for the protocol `pop3`.[2] The different markers correspond to the sources of the training data. Two of the

[2] Sensitivity is the proportion of positives detected out of all positives (proportion correctly verified protocols). Specificity is the proportion of true positives out of all classified as positives (proportion of true protocols out of all non-rejected protocols).

Table 2. Protocols and number of flows in the three data sets

		UNIBS		CAIDA		LBNL	
Protocol	Port	Training	Testing	Training	Testing	Training	Testing
pop3	110	19611	19940	9591	2386	1172	1426
smtp	25	19427	19480	11831	20722	20825	1304
http	80	7063	4928	5930	12459	81984	38228
ftp	21	6296	14458	1652	16202	–	–
BitTorrent	6881	5057	7412	–	–	–	–
msn	1863	1024	1033	–	–	–	–
netbios-ssn	139	–	–	–	–	4575	10113
https	443	–	–	25427	7896	18013	3283
oms	4662	–	–	1716	2491	–	–
imap4	993	–	–	–	–	7677	422
other		–	7912	–	1423	–	4584

Table 3. Classification accuracy of the ensemble member classifiers (cross-data training and testing)

	pop3			smtp			http		
	UNIBS	LBNL	CAIDA	UNIBS	LBNL	CAIDA	UNIBS	LBNL	CAIDA
UNIBS	95.69	99.32	75.57	98.75	99.01	96.67	97.93	88.14	97.75
LBNL	78.99	99.11	76.53	81.46	99.17	95.39	99.29	95.49	94.78
CAIDA	83.42	99.21	99.83	80.44	99.34	99.54	98.50	93.68	97.94

anomalies with a substantial slip in the classification accuracy are indicated. The reasons for the inadequate classification can be illustrated with the findings in Figures 2 and 3. The presentation of the pop3 protocol is very different from one network to another. In addition, the classification of pop3 is further impaired by its similarity to ftp. The two marked points are for ensembles trained on UNIBS and LBNL and tested on CAIDA. The pop3 protocol have similar appearance in the UNIBS and LBNL data and a different, more scattered, appearance in the CAIDA data (Figure 3 (b)). Thus the ensembles trained on UNIBS and LBNL data are ill-equipped to classify the version of pop3 in CAIDA.

Figure 5 plots Sensitivity versus 1 − Specificity for protocols smtp and http. There are two inaccurate classifiers for smtp. This time the mismatch is between the smtp traffic in the UNIBS and LBNL data. The two points that lie closer to the diagonal line in subplot (a) are the cross-testing UNIBS-LBNL and LBNL-UNIBS.

The experimental results show that the ensemble members have accuracies comparable to those in the state-of-the-art literature on traffic classification [1,6,15]. The high accuracy of the cross-data experiment, with a few exceptions discussed earlier, indicates that the statistical approach to traffic classification is robust across networks, so universal solutions can be sought. This reflects the fact that the network protocols have standard definitions, and the features we are using are not affected by differences in network configurations, traffic intensity or delays.

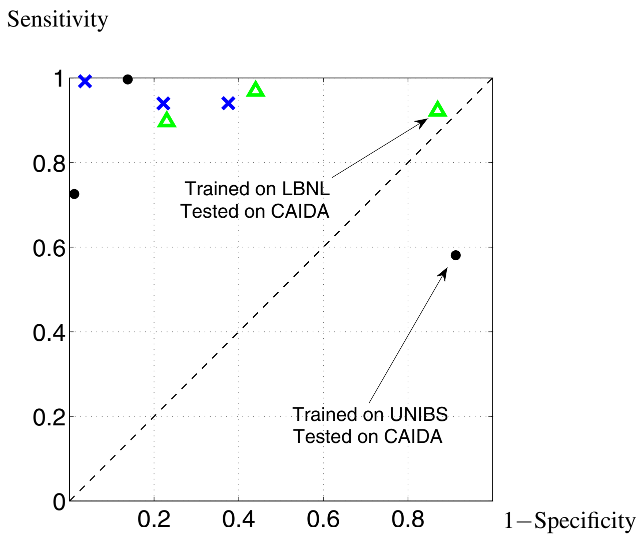


Fig. 4. Sensitivity-Specificity plot for protocol `pop3`. The markers indicate the training data source.

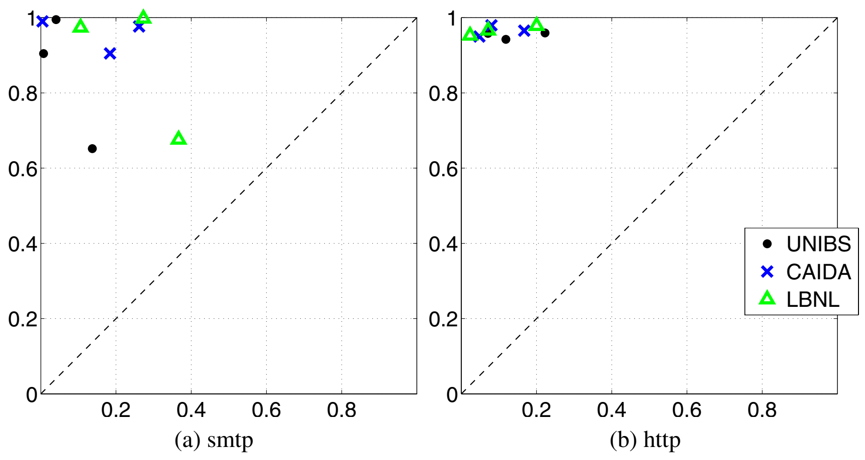


Fig. 5. Sensitivity versus 1−Specificity for protocols `smtp` and `http`

4 Conclusion

We propose a classifier selection ensemble for network traffic verification. Upon receiving the first four packets of a flow, the classifier selector directs the decision to an ensemble member based upon the requested port number. The classifier responsible for this port number ramifies the decision further using the "sign pattern" of the four packets. A decision tree classifier labels the flow as 'accepted' or 'rejected'. The 'accepted' class is the protocol conventionally associated with the requested port number. The flow can be classed as 'rejected' at every stage of the ensemble classification: the oracle rejects non-allowed port requests, the sign filter rejects flows whose sign patterns are highly unlikely, and, finally, the decision tree classifier is responsible for the fine discrimination based on the sizes of the first four packets. Without looking at the payloads, it is difficult to detect "tunnelling" behaviours where an unaccepted protocol is wrapped and carried within an accepted one. In our system, tunnelling may pass through the port and the sign filters but then land as an outlier in the space of the payload sizes. Our system is expected to reject the protocol at this final stage.

The classifiers were cross-tested using designated training and testing data of network traffic traces from three institutions: UNIBS, LBNL and CAIDA. The results show the robustness of the statistical approach to traffic classification.

While the ensemble accuracy is comparable to that reported in the literature [1,6,15] the proposed ensemble has the following advantages:

– Compared to classifiers that use the whole flow, our protocol verification is quick, as the sequence of decisions is based on the port number and the directions and sizes of the first four packets of a flow. Should operational speed permit it, the ensemble can be used online; a flow can be stopped before the application is processed by the network.
– The proposed ensemble needs only two parameters. First, we must choose a threshold for selecting the valid sign patterns (Here we used 2%. All sign patterns with likelihood higher than the threshold will merit separate decision tree classifiers. Flows with unlikely sign patterns are rejected.) The second parameter is the level of pruning for the decision tree classifiers.
– The structure of the ensemble is modular. Classifiers can be trained and added without disturbing the rest of the ensemble. For example, if a new port joins the list of allowed ports, an ensemble member can be trained separately for this port. Also, if the traffic changes, e.g., by allowing a new application through an existing protocol, and an unlikely sign pattern starts appearing more often, a separate classifier can be trained for this sign pattern and added to the ensemble.

One interesting future research direction comes from the fact that network traffic changes by definition, and so would the class descriptions ('accepted' and 'rejected') [10]. To respond to these changes, the ensemble should be further developed so as to cope with concept drift. Moreover, we are planning to implement our protocol verification system in an online platform such as the one described in [4].

References

1. Auld, T., Moore, A.W., Gull, S.F.: Bayesian neural networks for internet traffic classification. IEEE Trans. on Neural Networks 18(1), 223–239 (2007)
2. Bernaille, L., Teixeira, R., Salamatian, K.: Early application identification. In: CoNEXT 2006: Proceedings of the 2006 ACM CoNEXT conference, pp. 1–12. ACM, New York (2006)
3. Crotti, M., Dusi, M., Gringoli, F., Salgarelli, L.: Detecting http tunnels with statistical mechanisms. In: Proc. IEEE International Conference on Communications ICC 2007, pp. 6162–6168 (2007)

4. Dainotti, A., de Donato, W., Pescapè, A., Ventre, G.: Tie: a community-oriented traffic classification platform. Technical Report TR-DIS-10-2008, Dipartimento di Informatica e Sistemistica, University of Napoli Federico II (2008)
5. Dusi, M., Crotti, M., Gringoli, F., Salgarelli, L.: Detection of encrypted tunnels across network boundaries. In: Proc. IEEE International Conference on Communications ICC 2008, May 19–23, pp. 1738–1744 (2008)
6. Este, A., Gargiulo, F., Gringoli, F., Salgarelli, L., Sansone, C.: Pattern recognition approaches for classifying ip flows. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T.-Y., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSPR/SPR 2008. LNCS, vol. 5342, pp. 885–895. Springer, Heidelberg (2008)
7. Freire, E.P., Ziviani, A., Salles, R.M.: On metrics to distinguish skype flows from http traffic. In: Proc. Latin American Network Operations and Management Symposium LANOMS 2007, pp. 57–66 (2007)
8. Garner, S.R.: Weka: The waikato environment for knowledge analysis. In: Proc. of the New Zealand Computer Science Research Students Conference, pp. 57–64 (1995)
9. Holanda Filho, R., Fontenelle do Carmo, M.F., Maia, J., Siqueira, G.P.: An internet traffic classification methodology based on statistical discriminators. In: Proc. IEEE Network Operations and Management Symposium NOMS 2008, pp. 907–910 (2008)
10. Kuncheva, L.I.: Classifier ensembles for changing environments. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 1–15. Springer, Heidelberg (2004)
11. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
12. Li, Z., Yuan, R., Guan, X.: Traffic classification - towards accurate real time network applications. In: HCI, vol. (4), pp. 67–76 (2007)
13. Moore, D., Keys, K., Koga, R., Lagache, E., Claffy, K.C.: The coralreef software suite as a tool for system and network administrators. In: LISA 2001: Proceedings of the 15th USENIX conference on System administration, Berkeley, CA, USA, pp. 133–144. USENIX Association (2001)
14. Rastrigin, L.A., Erenstein, R.H.: Method of Collective Recognition. Energoizdat, Moscow (1981) (in Russian)
15. Risso, F., Baldi, M., Morandi, O., Baldini, A., Monclus, P.: Lightweight, payload-based traffic classification: An experimental evaluation. In: Proc. IEEE International Conference on Communications ICC 2008, pp. 5869–5875 (2008)
16. Williams, N., Zander, S., Armitage, G.: A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. SIGCOMM Comput. Commun. Rev. 36(5), 5–16 (2006)
17. Woods, K., Kegelmeyer, W.P., Bowyer, K.W.: Combination of multiple classifiers using local accuracy estimates. IEEE Trans. Pattern Anal. Mach. Intell. 19(4), 405–410 (1997)

Supervised Selective Combining Pattern Recognition Modalities and Its Application to Signature Verification by Fusing On-Line and Off-Line Kernels

Alexander Tatarchuk[1], Valentina Sulimova[2], David Windridge[3], Vadim Mottl[1], and Mikhail Lange[1]

[1] Computing Center of the Russian Academy of Sciences, Moscow, Russia
[2] Tula State University, Tula, Russia
[3] Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, UK

Abstract. We consider the problem of multi-modal pattern recognition under the assumption that a kernel-based approach is applicable within each particular modality. The Cartesian product of the linear spaces into which the respective kernels embed the output scales of single sensors is employed as an appropriate joint scale corresponding to the idea of combining modalities at the sensor level. This contrasts with the commonly adopted method of combining classifiers inferred from each specific modality. However, a significant risk in combining linear spaces is that of overfitting. To address this, we set out a stochastic method for encompassing modal-selectivity that is intrinsic to (that is to say, theoretically contiguous with) the selected kernel-based pattern-recognition approach.

The principle of kernel selectivity supervision is then applied to the problem of signature verification by fusing several on-line and off-line kernels into a complete training and verification technique.

1 Introduction

It is often appropriate to treat observed phenomena via several distinct feature modalities (frequently with differing measurement scales) for the purposes of pattern recognition [1,2]. Such feature scales $x_i \in \mathbb{X}_i$ may be such that it is convenient, or even necessary, to treat real-world objects $\omega \in \Omega$ via a pair-wise similarity measure over these features $\big(x_i(\omega'), x_i(\omega'')\big)$. It is therefore assumed that mode-specific functions $K_i(x_i', x_i'')$ can be delimited over the output scales of the sensors in question $\mathbb{X}_i \times \mathbb{X}_i \to \mathbb{R}$. The various $K(x', x'')$ functions constitute a *kernel* if they embed the sensor output $\mathbb{X}_i$ into a linear space via analogy with the inner-product. This condition is satisfied if the Kernel function defines a semidefinite matrix over any finite set of measured objects. The embedding may be of a significantly (even infinitely) different dimensionality to that of the original sensor scale, depending on the kernel characteristics.

Kernel-based multi-modal pattern recognition presents a number of difficulties and advantages over classical pattern-recognition in consequence of its pairwise nature. In particular, the problem of the composition and selection of feature modalities becomes acute, since we cannot simply assume the Euclidean vectorisablity of composite data without explicit construction of a kernel in the composite space. This problem is further compounded by the potential presence of training data that is not equally represented within each modality - as sometimes occurs in census returns, or in independently-trained classification systems, for example, in multimodal biometrics[1].

However, when $x_i(\omega) \in \mathbb{X}_i = \mathbb{R}$, the kernel defined by the product $K_i(x_i', x_i'') = x_i' x_i''$ generates an appropriate and natural embedding of the multimodal data.

The class of discriminative classifiers known as Support Vector Machines (SVMs) may thus be employed for two-class pattern recognition within $\mathbb{R}^n$, once modalities are combined via the joint kernel $K(\mathbf{x}', \mathbf{x}'') = \sum_{i=1}^{n} x_i' x_i''$ (this approach can also be used for highly-complex kernel-represented modalities [3,4,5]).

Despite the improved resilience of the SVM approach to over-fitting by virtue of its adjustment of capacity to the requirements of hyperplane description, it is often still necessary to combine modality-specific features only after selection has taken place. Feature selection (FS) techniques are of two broad types: *filters* and *wrappers* [6].

Filters are applied to the feature set irrespective of classification methodology, in contrast to wrappers. In this case, selection is either continuous (via weighting of the features) or else carried-out through absolute inclusion/exclusion of features from the total set. Wrappers, while considering feature selection in conjunction with classification, do not, in general, seek to do so via a single algorithmic approach (ie one in which FS is implicit in the process of classification itself - an exception being [7]). This is perhaps because of the danger of sample variability; if classification and FS progress interdependently, outliers can potentially affect the process disproportionately in the earlier stages. If, on the other hand, there exists a method of assigning selectivity *a priori*, this danger is mitigated to a large extent. Ideally, we require a range of behaviours, from the complete absence of selection, to the selection of only singular features.

In the following paper, we show, following [9] and [10], how selectivity may be incorporated into the Relevance Kernel Machine (RKM) [4,5], a continuous wrapper FS method previously described by the authors. The desired selectivity is achieved through a meta-parameter that controls the tendency of the RKM to generate zero components in the orientation of the decision plane (and hence the degree of elimination of constituent kernels). Thus, the selectivity parameter corresponds directly to model complexity, with the appropriate level of selectivity determined by cross validation or (in future work) via information-theoretic considerations.

[1] This missing data issue also occurs, albeit less acutely, in standard pattern recognition: the reason for its particularly problematic nature in kernel-based pattern-recognition is the inability to construct an embedding space when presented with an incomplete kernel Gram matrix w.r.t all of the measured objects.

The Relevance Kernel Machine with supervised selectivity is then applied to the problem of signature verification which consists in testing the hypothesis that a given signature belongs to the person having claimed his/her identity. Depending on the initial data representation, it is adopted to distinguish between on-line and off-line signature verification [8]. Any method of signature verification is based, finally, on a metric or kernel in the set of signatures. The selective kernel fusion technique considered in this paper serves as a natural way of easily combining on-line and off-line methods into an entire signature verification procedure. Experiments with signature database SVC2004 have shown that the multi-kernel approach essentially decreases the error rate in comparison with verification based on single kernels.

2 A Bayesian Strategy for Determining the Discriminant Hyperplane

Let objects $\omega \in \Omega$, measured by n features with modality-specific scales $x_i(\omega) \in \mathbb{X}_i$, be allocated to one of two classes $y(\omega) \in \mathbb{Y} = \{-1, 1\}$. For convenience, we assume an underlying distribution in the set of observable feature values and associated class indices; $\big(x_1(\omega), ..., x_n(\omega), y(\omega)\big) \in \mathbb{X}_1 \times ... \times \mathbb{X}_n \times \mathbb{Y}$. Training set members $(X, Y) = \{x_{1j}, ..., x_{nj}, y_j,\ j = 1, ..., N\}$, $x_{ij} = x_i(\omega_j)$, $y_j = y(\omega_j)$ are i.i.d. The kernel approach demands only that a real value similarity function exists - it thus obviates the distinction between different kinds of feature scales, so that we can assume that all the modality-specific features $x_i(\omega) \in \mathbb{X}_i$ are real-valued: $\mathbb{X}_i = \mathbb{R}$.

Functions $\varphi_1\left(x_1, ..., x_n \,|\, a_1, ..., a_n, b, y\right)$ with $y = \pm 1$ are thus two parametric families of probability densities in the composite feature space $\mathbb{X}_1 \times ... \times \mathbb{X}_n$. We assume marginally overlapping concentrations, such that the two together can be associated with a discriminant hyperplane $\sum_{i=1}^{n} a_i x_i + b \gtrless 0$. We further associate improper (ie non-unity integral) densities with the distributions:

$$\varphi\left(x_1, ..., x_n \,|\, a_1, ..., a_n, b, y\right) = \begin{cases} h, & y\left(\sum_{i=1}^{n} a_i x_i + b\right) > 1, \\ \exp\left[-c\big(1 - y\left(\sum_{i=1}^{n} a_i x_i + b\right)\big)\right], & y\left(\sum_{i=1}^{n} a_i x_i + b\right) < 1, \end{cases}$$

The constant h then represents the extent to which the classes are equivalent to a uniform distribution over their respective half-spaces. The parameter c determines the extent to which the classes overlap.

The direction vector $(a_1, ..., a_n)$ of the discriminant hyperplane $\sum_{i=1}^{n} a_i x_i + b \gtrless 0$ will, in the absence of a training mechanism, be considered a random vector distributed in accordance with some specific prior density $\Psi(a_1, ..., a_n \,|\, \mu)$ parametrized by μ. No such constraint is assumed in b, hence, $\Psi(a_1, ..., a_n, b \,|\, \mu) \propto \Psi(a_1, ..., a_n | \mu)$.

With respect to the training set, the *a posteriori* joint distribution density of the parameters of the discriminant hyperplane is consequently proportional to the product $P(a_1, ..., a_n, b \,|\, X, Y, \mu) \propto \Psi(a_1, ..., a_n \,|\, \mu) \times \Phi(X \,|\, Y, a_1, .., a_n, b)$.

The objective of training is thus to maximise the *a posteriori* density:

$$\begin{aligned}&(\hat{a}_1, ..., \hat{a}_n, \hat{b}) = \\ &\arg\max\left[\ln \Psi(a_1, ..., a_n \,|\, \mu) + \ln \Phi(X \,|\, Y, a_1, .., a_n, b)\right].\end{aligned}$$

This correlates to the training criterion:

$$\begin{cases} -\ln \Psi(a_1, ..., a_n|\mu) + c\sum_{j=1}^{N} \delta_j \to \min\limits_{(a_1,...,a_n,b,\delta_1,...,\delta_N)}, \\ y_j \left(\sum_{i=1}^{n} a_i x_{ij} + b\right) \geq 1 - \delta_j,\ \delta_j \geq 0,\ j = 1, ..., N. \end{cases} \tag{1}$$

Note that if we set $C = 2rc$, with r the common variance of the independent constituent variables (having zero mean), and omit the parameter μ (such that $\Psi(a_1, ..., a_n \,|\, \mu) = \Psi(a_1, ..., a_n)$ is the joint normal distribution), we obtain the classical SVM over the real-valued features $x_{ij} \in \mathbb{X}_i = \mathbb{R}$ with the direction vector elements $a_i \in \mathbb{X}_i = \mathbb{R}$ constituting a discriminant hyperplane in $\mathbb{X}_1 \times ... \times \mathbb{X}_n = \mathbb{R}^n$ such that:

$$\begin{cases} \sum_{i=1}^{n} a_i^2 + C\sum_{j=1}^{N} \delta_j \to \min\limits_{(a_1,...,a_n,b,\delta_1,...,\delta_N)}, \\ y_j \left(\sum_{i=1}^{n} a_i x_{ij} + b\right) \geq 1 - \delta_j,\ \delta_j \geq 0,\ j = 1, ..., N. \end{cases} \tag{2}$$

Specifically, if the kernels $K_i(x_i', x_i'') : \mathbb{X}_i \times \mathbb{X}_i \to \mathbb{R}$ defined for the sensor features $x_i \in \mathbb{X}_i$ are inserted into (2), we obtain the optimization:

$$\begin{cases} \sum_{i=1}^{n} K_i(a_i, a_i) + C\sum_{j=1}^{N} \delta_j \to \min\limits_{(a_1,...,a_n,b,\delta_1,...,\delta_N)}, \\ y_j \left(\sum_{i=1}^{n} K_i(a_i, x_{ij}) + b\right) \geq 1 - \delta_j,\ \delta_j \geq 0, \\ j = 1, ..., N. \end{cases} \tag{3}$$

It is important to note that, in general, the elements a_i of the hyperplane direction vector exist in the embedding space $\tilde{\mathbb{X}}_i \supseteq \mathbb{X}_i$, rather than the original feature space $\mathbb{X}_i$.

A central advantage of SVMs, in terms of their capacity for overfitting, is that at the minimum of the training criterion (such that $a_i = \sum_{j:\ \lambda_j > 0} \lambda_j y_j x_{ij} \in \tilde{\mathbb{X}}_i$), the discriminant hyperplane applicable to any new point $(x_i \in \mathbb{X}_i,\ i = 1, \ldots, n)$

$$\sum\nolimits_{j:\ \lambda_j > 0} \lambda_j y_j \sum\nolimits_{i=1}^{n} K_i(x_{ij}, x_i) + b \gtrless 0 \tag{4}$$

is determined only by those Lagrange multipliers with $\lambda_j \geq 0$ in the dual form of (3), ie the *support objects*. The dual problem, which can be solved by quadratic-programming is thus :

$$\begin{cases} \sum\limits_{j=1}^{N} \lambda_j - (1/2) \sum\limits_{j=1}^{N} \sum\limits_{l=1}^{N} y_j y_l \Big(\sum\limits_{i=1}^{n} K_i(x_{ij}, x_{il})\Big) \lambda_j \lambda_l \to \max, \\ \sum\limits_{j=1}^{N} y_j \lambda_j = 0,\ 0 \leq \lambda_j \leq C/2,\ j = 1, ..., N. \end{cases} \tag{5}$$

The following section will consider a distinct form of the *a priori* distribution $\Psi(a_1, ..., a_n \,|\, \mu)$, that gives rise to a feature- and kernel-selective SVM, such that the parameter μ controls the desired selectivity level.

3 The Continuous Training Technique with Supervised Selectivity

We first assume a conditional normal distribution for the direction elements a_i in relation to independent random variances given by r_i:

$$\begin{aligned}&\psi(a_i \,|\, r_i) = \left(1\Big/r_i^{1/2}(2\pi)^{1/2}\right)\exp\left(-(1/2r_i)a_i^2\right),\\ &\Psi(a_1,...,a_n \,|\, r_1,...,r_n) \propto\\ &\left(\prod_{i=1}^n r_i\right)^{-1/2}\exp\Big(-(1/2)\sum_{i=1}^n (1/r_i)a_i^2\Big).\end{aligned}$$

There is hence a hyper-ellipsoidal relationship between the direction elements a_i.

We further assume that the reciprocated variances are gamma distributed (a reasonable, maximum-entropy-based assumption for positive-constrained scale variables), ie: $\gamma\big((1/r_i) \,|\, \alpha, \beta\big) \propto (1/r_i)^{\alpha-1}\exp\left(-\beta\,(1/r_i)\right)$ (with means $E(1/r_i) = \alpha/\beta$ and variances $E\left((1/r_i)^2\right) = \alpha/\beta^2$). We then set the following parameter relations to enable convenient characterisation of the distribution; $\alpha = (1+\mu)^2/2\mu$, $\beta = 1/2\mu$.

There is hence now a parametrically-defined set of distributions in the direction elements a_i, dependant only on $\mu : \mu \geq 0$ (where $E(1/r_i) = (1+\mu)^2$ and $E\left((1/r_i)^2\right) = 2\mu(1+\mu)^2$).

In behavioral terms it should be noted that, as $\mu \to 0$, we find that $1/r_i \cong ... \cong 1/r_n \cong 1$. However, as μ increases, this identity constraint is progressively relaxed.

Proceeding with the derivation, we now eliminate the inverse variances as follows. Firstly, we note that the joint distribution of independent inverse variances with respect to μ is proportional to the product:

$$G(r_1,...,r_n \,|\, \mu) \propto \left(\prod_{i=1}^n (1/r_i)\right)^{(1+\mu)^2/2\mu-1}\exp\left(-1/2\mu\sum_{i=1}^n (1/r_i)\right).$$

The maximum of the joint *a posteriori* density function $P(a_1,...,a_n,b,r_1,...,r_n \,|X,Y,\mu)$ then gives us the required training criterion: we see that it is proportional to the product: $\Psi(a_1,...,a_n \,|\, r_1,...,r_n)\; G(r_1,...,r_n \,|\, \mu)\; \Phi(X \,|\, Y, a_1,..,a_n,b)$.

In the case of real-valued features $x_i \in \mathbb{R}$, the resulting training criterion hence has the form:

$$\begin{cases} \sum_{i=1}^n \left[(1/r_i)\left(a_i^2+(1/\mu)\right)+((1/\mu)+1+\mu)\ln r_i\right]+\\ \qquad C\sum_{j=1}^N \delta_j \to \min\left(a_i \in \mathbb{R}, r_i, b, \delta_j\right),\\ y_j\left(\sum_{i=1}^n a_i x_{ij}+b\right) \geq 1-\delta_j,\ \delta_j \geq 0,\ j=1,...,N,\\ \qquad r_i \geq \varepsilon, \end{cases} \tag{6}$$

$\varepsilon > 0$ is the inclusion criterion for features: it is thus a sufficiently small positive real number. In general, a smaller r_i will imply a smaller a_i. As $r_i \to \varepsilon$, the ith feature will affect the discriminant hyperplane $\sum_{i=1}^n a_i x_i + b \gtrless 0$ increasingly weakly.

Again, we obtain the kernel-based training criterion by substituting into (6) $K_i(a_i,a_i)$ for a_i^2 and replacing $a_i x_{ij}$ by $K_i(a_i,x_{ij})$ to give:

$$\begin{cases} \sum_{i=1}^{n}\left[(1/r_i)\left(K_i(a_i,a_i)+(1/\mu)\right)+\right. \\ \left.\left((1/\mu)+1+\mu\right)\ln r_i\right]+C\sum_{j=1}^{N}\delta_j \to \min\limits_{a_i\in\tilde{\mathbb{X}}_i, r_i, b, \delta_j}, \\ y_j\left(\sum_{i=1}^{n}K_i(a_i,x_{ij})+b\right)\geq 1-\delta_j, \delta_j\geq 0, j=1,\dots,N, r_i\geq\varepsilon. \end{cases} \tag{7}$$

As with SVMs, there is no explicit need to evaluate either the $a_i \in \mathbb{R}$ in (6) or the $a_i \in \tilde{\mathbb{X}}_i$ in (7); it is sufficient merely to establish the non-zero Lagrange multipliers $\lambda_j \geq 0$ in the dual representation $a_i = r_i \sum_{j:\,\lambda_j>0} y_j\lambda_j x_{ij}$. We do this via quadratic-programming using a modification of (5):

$$\begin{cases} \sum\limits_{j=1}^{N}\lambda_j - \frac{1}{2}\sum\limits_{j=1}^{N}\sum\limits_{l=1}^{N} y_j y_l\left(\sum\limits_{i=1}^{n} r_i K_i(x_{ij},x_{il})\right)\lambda_j\lambda_l \to \max, \\ \sum\limits_{j=1}^{N} y_j\lambda_j = 0,\ 0\leq\lambda_j\leq C/2,\ j=1,...,N. \end{cases} \tag{8}$$

This gives the Kernelised decision hyperplane:

$$\sum\nolimits_{j:\,\lambda_j>0} y_j\lambda_j \sum\nolimits_{i=1}^{n} r_i K_i(x_{ij},x_i)\ +\ b \gtrless 0 \tag{9}$$

In distinction to the discriminant hyperplanes for standard SVMs (4), features are effectively assigned weights r_i, so that as $r_i \to 0$, the influence of the respective features diminishes. However, as it stands, the weights are unknown in (7).

Solving this optimization problem for fixed μ, involves the application of the Gauss-Seidel iteration to the variable sets $(a_1,...,a_n,b,\delta_1,\dots,\delta_N)$ and $(r_1,...,r_n)$, with initiation values of $(r_i^0 = 1,\ i=1,...,n)$. Once the solution $\lambda_1^k,\dots,\lambda_N^k$, i.e. $(a_1^k,...,a_n^k)$, is found at the k th iteration with the current approximations $(r_1^k,\dots, r_n^k)$, the revised values of the variances $(r_1^{k+1},...,r_n^{k+1})$ are defined as

$$\begin{gathered} r_i^{k+1}=\tilde{r}_i^{k+1}\ \text{ if }\ \tilde{r}_i^{k+1}\geq\varepsilon,\ r_i^{k+1}=\varepsilon\ \text{ otherwise}, \\ \tilde{r}_i^{k+1}=\frac{(a_i^k)^2+1/\mu}{1/\mu+1+\mu}= \\ \frac{\sum_{j:\lambda_j^k>0}\sum_{l:\lambda_l^k>0} y_j y_l\,(r_i^k)^2 K_i(x_{ij},x_{il})\lambda_j^k\lambda_l^k+1/\mu}{1/\mu+1+\mu}. \end{gathered} \tag{10}$$

Convergence of the procedure occurs in $\approx 10-15$ steps for typical problems, suppressing redundant features through the allocating of very small (but always non-zero weights) r_i defining the discriminant hyperplane (9).

In summary, the training criterion for Relevance Kernel Machine (RKM) [4,5] is set out in (6). The feature selectivity of this SVM generalisation is parametrically determined by $\mu : 0 \leq \mu < \infty$. As $\mu \to 0$, variances tend toward unity (10), and the RKM degenerates to the classical SVM (2). Contrarily, when $\mu \to \infty$, we have from (6) that $\sum_{i=1}^{n}\left[(1/r_i)a_i^2+(1+\mu)\ln r_i\right]+C\sum_{j=1}^{N}\delta_j \to \min$; actually a significantly more selective training criterion than the original RKM (without supervised selectivity): $\sum_{i=1}^{n}\left[(1/r_i)a_i^2 + \ln r_i\right] + C\sum_{j=1}^{N}\delta_j \to \min$ [4].

4 Signature Verification via Selective Fusion of On-Line and Off-Line Kernels

4.1 Kernels Produced by Metrics

Let ω' and ω'' be two signatures represented by signals or images, and $\rho(\omega', \omega'')$ be a metric evaluating dissimilarity of signatures from a specific point of view. Then function

$$K(\omega', \omega'') = \exp\left[-\gamma\, \rho^2(\omega', \omega'')\right] \tag{11}$$

has the sense of their pair-wise similarity. If coefficient $\gamma > 0$ is large enough, this function will be a kernel in the set of signatures, usually called the radial kernel.

As a rule, it is impossible to know in advance which of possible metrics is more appropriate for a concrete person. The advantages of the multi-kernel approach to the problem of on-line signature verification were demonstrated in [4]. We extend here the kernel-based approach onto the problem of combining the on-line and off-line modalities (Figure 1) into an entire signature verification technique.

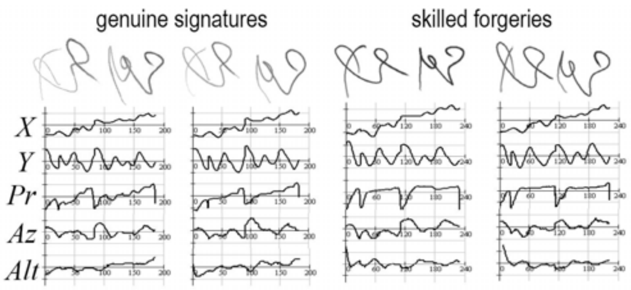


Fig. 1. Off-line (images) and on-line (signals) representation of signatures

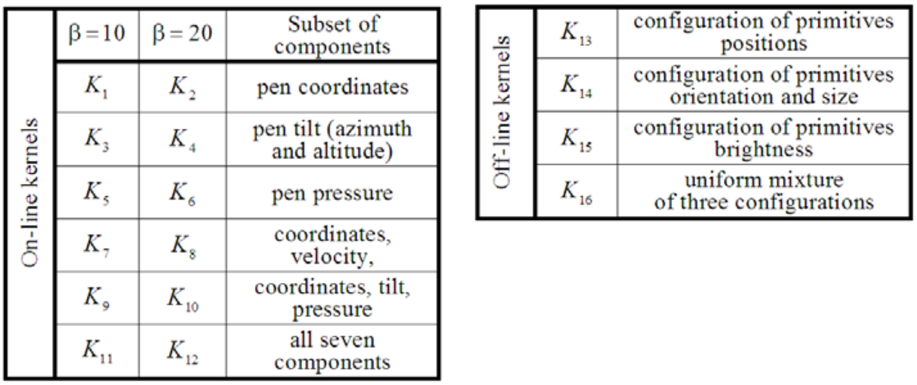
In this work, we tested 12 different metrics in the set of on-line signatures and 4 metrics computed from the pictorial off-line representation. So, all in all, we combined 16 different on-line and off-line kernels listed in Table 1.

4.2 Metrics in the Set of On-Line Signatures

Each on-line signature is represented by a multi-component vector signal which initially includes five components $\mathbf{x}_t = (x_t^1 \cdots x_t^n)$: two pen tip coordinates (X, Y), pen tilt azimuth (Az) and altitude (Alt), and pen pressure (Pr) (Fig. 1). We supplement the signals with two additional variables - pen's velocity and acceleration.

For comparing pairs of signals of different lengths $[\omega' = (\mathbf{x}'_s, s = 1, \ldots, N')$, $\omega'' = (\mathbf{x}''_s, s = 1, \ldots, N'')]$, we use the principle of dynamic time warping with the purpose of aligning the vector sequences [4]. Each version of alignment

Table 1. The kernels studied in the experiments



	β=10	β=20	Subset of components
On-line kernels	K_1	K_2	pen coordinates
	K_3	K_4	pen tilt (azimuth and altitude)
	K_5	K_6	pen pressure
	K_7	K_8	coordinates, velocity,
	K_9	K_{10}	coordinates, tilt, pressure
	K_{11}	K_{12}	all seven components

Off-line kernels	K_{13}	configuration of primitives positions
	K_{14}	configuration of primitives orientation and size
	K_{15}	configuration of primitives brightness
	K_{16}	uniform mixture of three configurations

$w(\omega', \omega'')$ is equivalent to a renumbering of the elements in both sequences $\omega'_w = (\mathbf{x}'_{w,s'_k},\, k = 1, \ldots, N_w)$, $\omega''_w = (\mathbf{x}''_{w,s''_k},\, k = 1, \ldots, N_w)$, $N_w \geq N'$, $N_w \geq N''$. We tested 12 different metrics defined by 6 different subsets of signal components and 2 different values of the alignment rigidity parameter β [4] as shown in Table 1:

$$\rho(\omega', \omega''|\beta) = \min_w \sqrt{\sum\nolimits_{k=1}^{N_w} \|\mathbf{x}'_{w,s'_k} - \mathbf{x}''_{w,s''_k}\|^2}. \tag{12}$$

4.3 Metrics in the Set of Off-Line Signatures

For comparing grayscale images (patterns) representing off-line signatures we apply the technique of tree-structured pattern representation proposed in [11].

For the given pattern P, the recursive scheme described in [11] produces a pattern representation R in the form of a complete binary tree of elliptic primitives (nodes) Q: $R = \{Q_n : 0 \leq n \leq n_{\max}\}$, where n is the node number of the level $l_n = \lfloor \log_2(n = 1) \rfloor$.

Let R' and R'' be a pair of tree-structured representations, and $R' \bigcap R''$ be their intersection formed by the pairs of nodes (Q'_n, Q''_n) having the same number n. For comparing any two corresponding nodes $Q'_n \in R'$ and $Q''_n \in R''$, a dissimilarity function $d(Q'_n, Q''_n) \geq 0$ can be easily defined through parameters of each primitive such as center vector, orientation vectors with their sizes (along two principal axes of the primitive), and the mean brightness value. Using these parameters, we define a loss function

$$D(Q'_n, Q''_n) = \begin{cases} d(Q'_n, Q''_n), \text{ if } Q'_n \text{ and/or } Q''_n \text{ are "end" nodes,} \\ 0, \text{ otherwise,} \end{cases}$$

where $d(Q'_n, Q''_n) = \alpha_1 d_1(Q'_n, Q''_n) + \alpha_2 d_2(Q'_n, Q''_n) + \alpha_3 d_3(Q'_n, Q''_n)$, $\alpha_1, \alpha_2, \alpha_3 \geq 0$, $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Here, $d_i(Q'_n, Q''_n)$ is a distinction function between the centers of the primitives, their orientation and size parameters, and the mean brightness values for $i = 1, 2, 3$, respectively.

Then, following [11], we define the distinction measure (metric) of the trees R' and R'' as follows:

$$\begin{aligned}&\rho(R', R'' \mid \alpha_1, \alpha_2, \alpha_3) = \sum\nolimits_{R' \bigcap R''} 2^{-l_n} D(Q'_n, Q''_n) = \\ &\alpha_1 \sum\nolimits_{R' \bigcap R''} 2^{-l_n} d_1(Q'_n, Q''_n) + \\ &\alpha_2 \sum\nolimits_{R' \bigcap R''} 2^{-l_n} d_2(Q'_n, Q''_n) + \\ &\alpha_3 \sum\nolimits_{R' \bigcap R''} 2^{-l_n} d_3(Q'_n, Q''_n),\end{aligned} \tag{13}$$

where the sum is taken over all pairs $(Q'_n, Q''_n) \in R' \bigcap R''$. We competitively applied three basic distinction measures of the form (13) $\rho_1(R', R'') = \rho(R', R'' \mid 1, 0, 0)$, $\rho_2(R', R'') = \rho(R', R'' \mid 0, 1, 0)$, $\rho_3(R', R'') = \rho(R', R'' \mid 0, 0, 1)$, and the uniform mixture $\rho_4(R', R'') = \rho(R', R'' \mid 1/3, 1/3, 1/3)$.

4.4 Signature Database and Results of Experiments

In the experiment, we used the database of the Signature Verification Competition 2004 [12] that contains vector signals of 40 persons (Fig. 1). On the basis of these signals we generated grayscale images (256×256 pixels) with 256 levels of brightness corresponding to the levels of pen pressure in the original signals.

For each person, the training set consists of 400 signatures, namely, 5 signatures of the respective person, 5 skilled forgeries, and 390 random forgeries

Table 2. Error rates for single kernels versus kernel fusion

Individual kernels		Error rate, %	Relevant as result of selective fusion for:
On-line kernels	K_1	0.507	5 persons
	K_2	0.870	10 persons
	K_3	5.543	0 persons
	K_4	7.500	0 persons
	K_5	2.750	0 persons
	K_6	2.500	4 persons
	K_7	0.870	2 persons
	K_8	1.304	1 person
On-line kernels	K_9	0.326	4 persons
	K_{10}	0.725	2 persons
	K_{11}	0.435	1 person
	K_{12}	1.015	0 persons
Off-line kernels	K_{13}	19.239	0 persons
	K_{14}	2.464	0 persons
	K_{15}	3.515	0 persons
	K_{16}	1.594	11 persons
Plain fusion		0.471	
Selective fusion		0.254	

formed by 195 original signatures of other 39 persons and 195 skilled forgeries for them. The test set for each person consists of 69 signatures, namely, 15 genuine signatures, 15 skilled forgeries, and 39 random forgeries. Thus, the total number of the test signatures for 40 persons amounts to 2760.

For each pair of signature signals, 12 different on-line metrics and 4 off-line metrics were simultaneously computed and, respectively, 16 different kernels were evaluated (Table 1).

For each person, we tested 18 ways of training based, first, on each of the initial kernels separately $\{K_1(\omega', \omega''), \ldots, K_{16}(\omega', \omega'')\}$, second, on the plane fusion of all the individual kernels with equal weights $(1/16)\sum_{i=1}^{16} K_i(\omega', \omega'')$, and, third, on the selective fusion of all the 16 kernels using the continuous training technique (Section 3) with the selectivity level chosen via cross validation. The error rates in the total test set of 2760 signatures are shown in Table 2.

It is well seen that the combined kernel obtained by selective kernel fusion with individually chosen selectivity essentially outperforms each of the single ones. At the same time, for each of 40 persons whose signatures made the data set, the kernel fusion procedure has selected only one relevant kernel as the most adequate representation of his/her handwriting.

5 Conclusions

The kernel-based approach to signature verification enables harnessing the kernel-selective SVM as one of mathematically most advanced methods of pattern recognition. This approach predefines the algorithms of both training and recognition, and it remains only to choose the kernel produced by an appropriate metric in the set of signatures, such that the genuine signatures of the same person would be much closer to each other than those of different persons. However, different understandings of signature similarity lead to different kernels.

The proposed kernel fusion technique automatically chooses the most appropriate subset of kernels for each person in the process of adaptive training. Experiments with signature data base SVC2004 demonstrate that verification results obtained by selective fusion of several on-line and off-line kernels in accordance with the proposed approach essentially outperforms the results based on both single kernels and their plane fusion.

Acknowledgements

This work was supported by the European Community's Seventh Framework Programme FP7/2007-2013 under Grant agreement No. 215078, the Russian Foundation for Basic Research (Grants 05-01-00679, 06-01-08042, 08-01-00695, 09-01-00573), INTAS Grant YSF-06-1000014-6563.

References

1. Ross, A., Jain, A.K.: Multimodal biometrics: An overview. In: Proceedings of the 12th European Signal Processing Conference (EUSIPCO), Vienna, Austria, pp. 1221–1224 (2004)

2. Jannin, P., Fleig, O.J., Seigneuret, E., Grova, C., Morandi, X., Scarabin, J.M.: A data fusion environment for multimodal and multi-informational neuronavigation. Computer Aided Surgery 5(1), 1–10 (2000)
3. Sonnenburg, S., Rätsch, G., Schäfer, C.: A general and efficient multiple kernel learning algorithm. In: Proceedings of the 19th Annual Conference on Neural Information Processing Systems, Vancouver, Canada, December 5-8 (2005)
4. Sulimova, V., Mottl, V., Tatarchuk, A.: Multi-kernel approach to on-line signature verification. In: Proceedings of the 8th IASTED International Conference on Signal and Image Processing, Honolulu, Hawaii, USA, August 14-16 (2006)
5. Mottl, V., Tatarchuk, A., Sulimova, V., Krasotkina, O., Seredin, O.: Combining pattern recognition modalities at the sensor level via kernel fusion. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 1–12. Springer, Heidelberg (2007)
6. Guyon, I.M., Gunn, S.R., Nikravesh, M., Zadeh, L. (eds.): Feature Extraction, Foundations and Applications. Springer, Heidelberg (2006)
7. Li, J., Zha, H.: Simultaneous classification and feature clustering using discriminant vector quantization with applications to microarray data analysis. In: Proceedings of the IEEE Computer Society Bioinformatics Conference, Palo Alto, CA, August 14-16, pp. 246–255 (2002)
8. Plamondon, R., Srihari, S.N.: On-line and off-line handwriting recognition: A comprehensive survey. IEEE Trans. on Pattern Recognition and Machine Intelligence 22(1), 63–84 (2000)
9. Tatarchuk, A., Mottl, V., Eliseyev, A., Windridge, D.: Selectivity supervision in combining pattern-recognition modalities by feature- and kernel-selective Support Vector Machines. In: Proceedings of the 19th International Conference on Pattern Recognition, Tampa, USA, December 8-11 (2008)
10. Mottl, V., Lange, M., Sulimova, V., Ermakov, A.: Signature verification based on fusion of on-line and off-line kernels. In: Proceedings of the 19th International Conference on Pattern Recognition, Tampa, USA, December 8-11 (2008)
11. Lange, M., Ganebnykh, S., Lange, A.: Moment-based pattern representation using shape and grayscale features. In: Martí, J., Benedí, J.M., Mendonça, A.M., Serrat, J. (eds.) IbPRIA 2007. LNCS, vol. 4477, pp. 523–530. Springer, Heidelberg (2007)
12. SVC 2004: First International Signature Verification Competition, `http://www.cs.ust.hk/svc2004/index.html`

Improved Uniformity Enforcement in Stochastic Discrimination

Matthew Prior and Terry Windeatt

Centre for Vision Speech and Signal Processing
University of Surrey, Guildford, Surrey, GU2 7XH, UK
{m.prior,t.windeatt}@surrey.ac.uk

Abstract. There are a variety of methods for inducing predictive systems from observed data. Many of these methods fall into the field of study of machine learning. Some of the most effective algorithms in this domain succeed by combining a number of distinct predictive elements to form what can be described as a type of committee. Well known examples of such algorithms are AdaBoost, bagging and random forests. Stochastic discrimination is a committee-forming algorithm that attempts to combine a large number of relatively simple predictive elements in an effort to achieve a high degree of accuracy. A key element of the success of this technique is that its coverage of the observed feature space should be uniform in nature. We introduce a new uniformity enforcement method, which on benchmark datasets, leads to greater predictive efficiency than the currently published method.

1 Introduction

There are many techniques available for inducing predictive algorithms from observed data. Those methods that use a combination of classifiers, called a committee or ensemble, such as AdaBoost[5], bagging[2] and random forests[7] have demonstrated very good performance on real-world problems. Stochastic discrimination is an alternative method of constructing committees of classifiers. It has a sound theoretical basis and is robust to sources of over-fitting, other than those attributable to small sample size effects[4, 8, 10]. It intrinsically deals with two class problems but can be extended to multi-class problems by the use of such techniques as one-versus-all and error correcting output coding[11] decompositions.

One of the principle differences between conventional ensemble methods, such as bagging, and stochastic discrimination is that in conventional ensembles each individual classifier is normally expert, to some degree, on the whole data space. In a stochastic discrimination ensemble this is not the case[9]. The set of weak classifiers in a stochastic discrimination ensemble may view the data space in a uniform fashion but individual classifiers may not, and in general will not, do this. More specifically, they will only consider a limited subspace of the feature space, with each dimension in the feature space having a degree of coverage selected at random.

The extent to which a stochastic discrimination ensemble views the feature space without unduly favouring one region over another is known as its uniformity. The method of uniformity enforcement is one of the key elements of an implementation of stochastic discrimination. Uniformity ensures that the ensemble as a whole can generalise effectively over the full extent of the feature space.

The implementation of stochastic discrimination described in detail in [9, 12] uses a method of uniformity enforcement that is based on measurements relating to the average coverage for elements of the ensemble predicting a specific class. We propose an alternative uniformity enforcement scheme based on the minimum instance coverage.

2 Stochastic Discrimination

Typically classifier combination methods such as AdaBoost, bagging and random forests seek to merge base classifiers that have knowledge of the full extent of the feature space via the training set. Stochastic discrimination differs in its approach to combining weak base classifiers, which it refers to as thick models, in that it seeks to assemble elements that cover subsets of the training data. These are drawn from embedded subspaces of a finite n dimensional space, $F \in \mathbb{R}^n$. These subspaces are constructed from a geometric model centered on an instance of the training set and which form an n dimensional rectangular parallelepiped.

The coverage of the parallelepiped in each feature dimension is a random proportion of the feature extent. This sub-sampling of the feature space is one of the methods responsible for ensuring diversity in the produced population of thick models and additionally acts as a regularisation mechanism to alleviate the potential for over-fitting.

A stream of thick models is generated by randomly selecting an instance from the training set and generating a geometric model around it. This stream is then

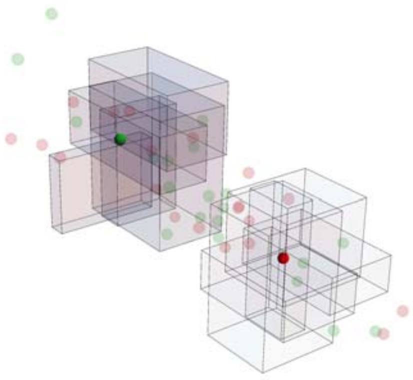


Fig. 1. A collection of 10 rectangular parallelepiped thick models centred around two data instances from the training set. Additional instances from the two classes are shown and they are all embedded in a three dimensional feature space.

Algorithm 1. Stochastic discrimination thick model stream production algorithm, P

```
Do
 Generate a Thick Model from a random instance in $TR_1$
 If ( Enriched( Thick Model ) )
     If( ImprovesUniformity( Thick Model ) )
           Accept( Thick Model )
Until ( Enough( Thick Models ) )
```

thinned according to each model's ability to discriminate between instances of classes within its embedded subspace of the feature domain. Suitably discriminating models undergo further selection based on the existing coverage of the feature domain. The underlying theory of stochastic discrimination[9] requires that coverage should be uniform to ensure that there is no bias towards particular areas of the feature space. This implies that the number of thick models capturing each instance in the training set should be equal.

In the context of a two class classification problem in which feature vectors, q, are drawn from two classes, $\{1, 2\}$, embedded in feature space F, instances from the available dataset are randomly partitioned into a training and test set, $\{TR, TE\}$. TR is further partitioned into instances from class 1 and class 2, $\{TR_1, TR_2\}$. Stochastic discrimination creates a stream of models by randomly sampling instances from TR and builds a space-enveloping thick model around them. The thick model, m, is constructed from random proportions of the feature extent in each of the n dimensions of the feature space, as depicted in Figure 1.

It is worthwhile observing that the generalisation ability of the stochastic discrimination algorithm is a function sensitive to a number of factors

$$SD_{Acc} = f(Model_{Number}, Enrichment_{Degree}, Coverage_{Uniformity}, Model_{Size}).$$

It is resistant to overtraining and in general the more models in the ensemble the higher the accuracy. Additionally there is a strong relationship between the number and distribution of instances in TR and the number of thick models required to adequately capture their distribution. Stochastic discrimination also relies on the assumption, which is a requirement for other machine learning algorithms, that there is a projectability between the distribution of samples in the training set and the test set.

2.1 Enrichment

For the stream of stochastically generated thick models to be useful for the task of classification it is necessary that they possess some discriminative power to separate the classes. To this end the thick models are selectively filtered based on their enrichment. Enrichment is calculated from the proportion of instances for each of the classes in the training set which are captured by the thick model, m.

If the proportion of instances of class 1 captured from TR_1 is greater than those of class 2 that are captured from TR_2, then the model is considered enriched with respect to class 1.

$$\frac{\mid m \subset TR_1 \mid}{\mid TR_1 \mid} > \frac{\mid m \subset TR_2 \mid}{\mid TR_2 \mid}. \tag{1}$$

2.2 Uniformity Enforcement Strategies

In the standard version of stochastic discrimination[9] each thick model subset, m, that satisfies the enrichment criteria is further subjected to a uniformity forcing step as indicated in Algorithm1 and referred as algorithm P. Uniformity is enforced via the measurement of coverage. The coverage of a data instance, c_q, is defined as the number of thick models that include the data instance, q, within their volume, divided by the size of the set of thick models, M, produced so far, $\mid M \mid$. Thus,

$$c_q = \frac{\mid \{m \in M : q \in m\} \mid}{\mid M \mid}. \tag{2}$$

A thick model, m, is considered an acceptable candidate for the ensemble of thick models if it is enriched and its average coverage for points captured from TR_1, c_{TR_1}, is below the average cover for all points in TR_1, $\bar{C}_{TR_1}$.

$$\Big(\frac{1}{|\{q \in TR_1 : q \in m\}|} \sum_{\{q \in TR_1 : q \in m\}} c_q\Big) < \bar{C}_{TR_1}. \tag{3}$$

In effect this filters models from the stream that favour instances which are under-represented in the current thick model working set, M.

We propose an alternative strategy to achieve uniformity. This entails selecting the least covered instance in TR_1. By choosing the instance that has minimum coverage as the basis for the new thick model, we aggressively focus on the area in the feature space that instantaneously exhibits the least coverage and ensure that it is increased. If there are ties for the least covered instance then these can be broken randomly or, as an enhancement, an instance from particularly ill represented region can be searched for.

$$\{q : \arg \min_{q \in TR_1} c_q\}. \tag{4}$$

Our modified algorithm, described in Algorithm 2 and referred to as L, seeks to improve the mean coverage and improve the variance of the coverage array by decreasing the contribution from the largest reducible component. If the minimum value in the coverage array, C, is not unique then one of the corresponding instances is selected at random.

To quantify the degree of uniformity, D, present in the thick model streams we calculate the standard deviation of the instance wise coverage.

$$D = \sqrt{\frac{1}{|TR|} \sum_{q \in TR} (c_q - \bar{C})^2}, \tag{5}$$

Algorithm 2. Stochastic discrimination thick model stream production algorithm, L

```
Do
 Generate a Thick Model using least covered instance in $TR_1$
 If ( Enriched( Thick Model ) )
          Accept( Thick Model )
Until ( Enough( Thick Models ) )
```

where $\overline{C}$ is the mean value of the coverage set. The minimum value for D is zero and this indicates that all instances in the training set have equivalent coverage, higher values of D represent increasingly poor levels of uniformity.

2.3 Discriminant

Once a thick model set of the desired size has been formed, a discriminant function can be used to classify instances. The discriminant function uses the difference in probabilities of capture by the thick models in M to assign class membership. In the case where the models have been enriched for TR_1, an unknown instance, x, will be captured by a larger number of thick models if it is of class 1 than were it of class 2. A suitable threshold can be chosen to optimise the classification accuracy of the discriminant function.

3 Theoretical Exploration

Under the assumption that variance is a valid measurement of uniformity we examine the behaviour of the classic uniformity algorithm P as defined in section 2.2 and Algorithm 1. and L Algorithm 2. These algorithms have simplified to highlight the essential differences between the two approaches. For more detailed implementation information see [9].

The coverage values, C, form a set of positive integers in the range from 0 to the size of the thick model set , $\mid M \mid$. Considering the limiting case in which a thick model captures only one point from TR, the addition of this thick model to M will result in a unity increment of a single value within C. The maximum reduction to the variance of C will be achieved if that point is the one that is the furthest below the mean value of the coverage set, $\overline{C}$, known as c_{min}. Ties in the value of c_{min} should be broken randomly.

By considering the contribution made to the change in variance by incrementing either c_{min} or another arbitrary member of C with a value larger than c_{min} we can show that

$$\frac{((k+1)-\overline{C})^2+(l-\overline{C})^2}{\mid M \mid} \leq \frac{((l+1)-\overline{C})^2+(k-\overline{C})^2}{\mid M \mid}, \qquad (6)$$

where k, l are positive real integers representing values within C and with $k \leq l$. It follows that the reduction in variance , and hence increase in the uniformity, will be greatest if k is the minimum value in C. The change to the mean value of C is constant in this limiting case. Thus for the special case where the thick model covers only a single instance in TR, algorithm L should always improve uniformity by at least as much as algorithm P.

When the thick model covers more than one point, the analysis of performance is more complicated. The degree of improvement of uniformity and the mean value of cover,$\overline{C}$, will depend on the specific sample of instances captured by the thick model, m. At the limit, where all points in TR_1 are captured by m, there will no difference in the change in uniformity and $\overline{C}$ between algorithms P and L. Where m only captures a percentage of TR_1 and if points are chosen at random, the expectation will be that the reduction in variance from algorithm L will always exceed or equal that from algorithm P. But this ignores the contribution from the average cover related enforcement strategy employed by algorithm P, which will undoubtedly improve the situation over a purely random selection. Furthermore, the performance will be dependent on the exact distribution of the dataset under consideration.

However, each new thick model under algorithm L will always contain the most beneficial point, c_{min}, whilst under algorithm P, m will only have some probability of capturing c_{min}. This probability will be dependent on the size of TR_1, the amount of the feature space that m captures and the distribution of instances in the feature space. Our experimental results suggest that on average algorithm L is more effective.

4 Experiment Details

Experiments were performed on twenty datasets, eighteen datasets from the UCI Machine Learning Repository [1] and 2 synthetic ones from [3]. These contained a mixture of binary and multi-class problems. Multi-class problems are handled using a one-versus-all decomposition strategy. To estimate the generalisation error of the induced classifiers, ten repetitions of ten-fold cross validation were performed for each dataset within a WEKA[6] framework. Identical trials were performed for the standard uniformity enforcement algorithm, described in [9], based on mean class coverages and our method of uniformity enforcement using the least covered point. The number of thick models used for each classifier was fixed at 3001. The minimum allowable thick model size was adjusted between 0.01 and 0.5 percent of the feature space. The following datasets were used. Balance[BAL], credit-a[CRA], diabetes[DIA], ecoli[ECO], glass[GLA], heart[HRT], hepatitis[HEP], ionosphere[ION], iris[IRS], labor[LAB], lymph[LYM], parkinsons[PAR], satellite[SAT], segment[SEG], sonar[SON], vehicle[VEH], vowel[VOW], Wisconsin breast cancer[WIS], twonorm[2NM] and threenorm[3NM].

5 Experimental Results

We present individual experimental results for a selection of the datasets in Figure 2. These show the test error rates and normalised standard deviation of the coverage, D, plotted against minimum model size for algorithms L and P. From Figure 2. it is not easy to determine a direct relationship between test accuracy and D, the trend, except in the case of WIS, is that lower coverage deviation leads to lower test error. The averaged values Figure 3. support this view. The minimum error rates in Table 1. confirm that neither algorithm is

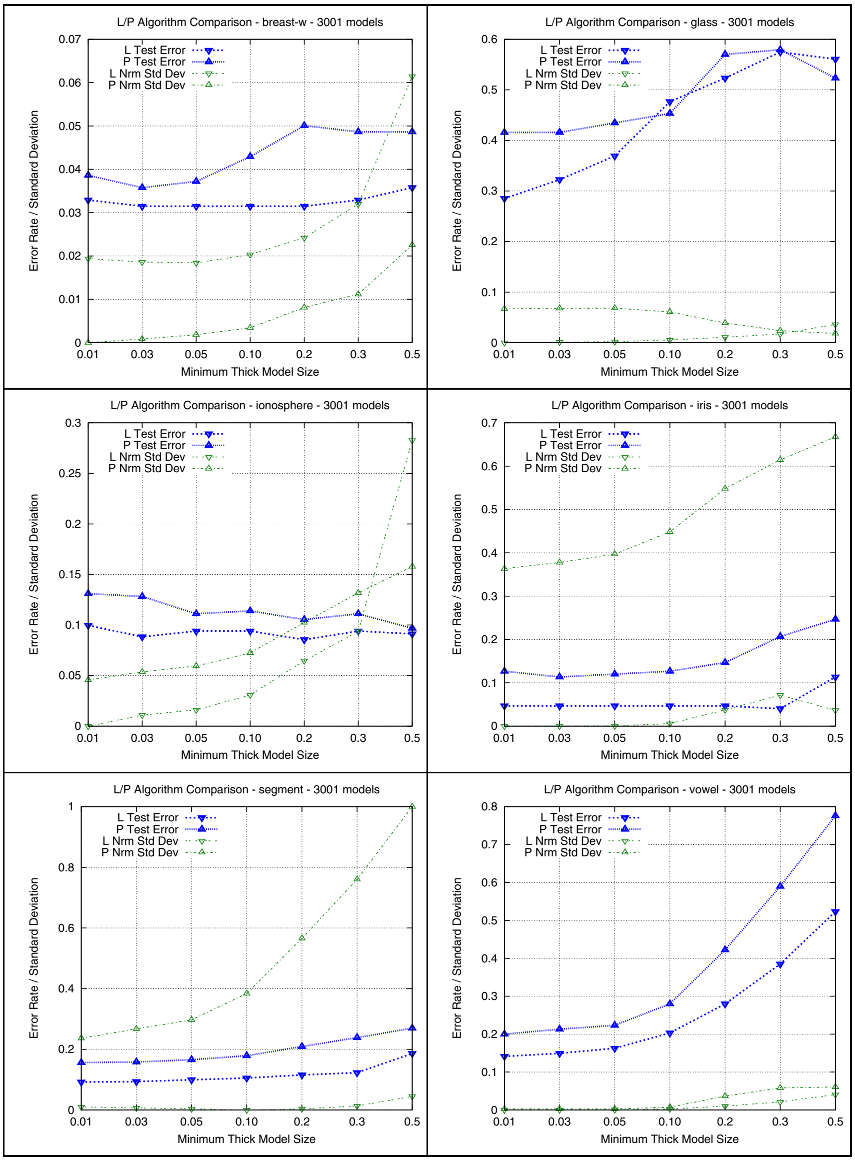


Fig. 2. Test error versus minimum thick model size and normalised standard deviation of coverage for uniformity enforcement algorithms L and P

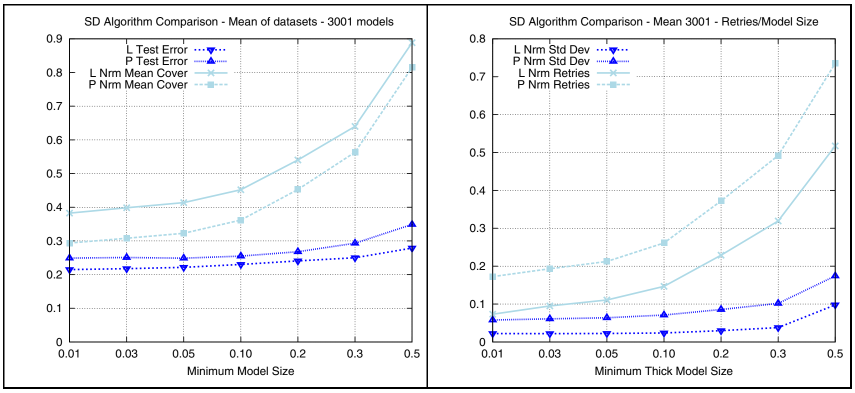


Fig. 3. Mean test error rate and normalised coverage over all datasets (left) and mean normalised retries and normalised standard deviation of coverage, D, (right)

Table 1. Minimum test error rates for algorithms L and P and mean values for L and P

	BAL	CRA	DIA	ECO	GLA	HRT	HEP	ION	IRS	LAB
L	.10	.03	.24	.14	.29	.17	.23	.09	.04	.19
P	.10	.04	.23	.16	.42	.17	.21	.10	.11	.19

	LYM	PAR	SAT	SEG	SON	VEH	VOW	WIS	2NM	3NM
L	.35	0.17	.12	.09	.12	.29	.14	.03	.03	.16
P	.47	0.13	.11	.16	.13	.32	.20	.04	.03	.14

Mean test error L	0.151
Mean test error P	0.173

superior on all datasets. Where L performs worse than P then the difference is generally small, as is the case with datasets DIA, HEP and SAT. Where algorithm L exceeds the performance of P the difference can be significant, as is the case with datasets GLA, IRS, LYM, SEG, VEH, VOW and the exceptions being PAR, 3NM.

Figure 3. contains averaged results over all datasets for test error and average cover on the left and normalised thick model retry rates and the normalised standard deviation of the coverage on the right. The averaged graphs give a clearer indication of the relative performance of the two algorithms. L consistently outperforms P in terms of test accuracy across all minimum thick model sizes and also for absolute coverage values. This implies that L is building larger thick models that capture more points and will tend to generalise better.

The right of Figure 3. shows that the averaged normalised standard deviation, D, is consistently better for L across all model sizes. It also shows that the stream production efficiency for L, measured by the number of retries required to find a suitable model, can be as little as half the value of P.

Table 1. shows the minimum value of the test error for each of the twenty datasets for uniformity enforcement algorithms L and P. Averaging over all datasets, algorithm P has a minimum test error that is 15 percent worse than L. Subjecting these results to a paired T test rejects the null hypothesis at a significance level of 0.05.

6 Conclusion

The strategy of uniform coverage enforcement is an important element of the stochastic discrimination method. Our experiments indicate that simply selecting the least covered instance in the training set is an effective alternative to the standard method of choosing a random instance and then checking for its effect on coverage. Though it is not certain for any particular dataset which strategy will be most effective, over a range of datasets, algorithm L achieves better accuracy, more uniform coverage, larger thick models and a lower retry rate than algorithm P. Finally, we would like to thank the reviewers for their helpful comments.

References

[1] Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)

[2] Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)

[3] Breiman, L.: Bias, variance, and arcing classifiers (1996)

[4] Chen, D., Huang, P., Cheng, X.: A concrete statistical realization of kleinberg's stochastic discrimination for pattern recognition, part i. two-class classification. Annals of Statistics 31(5), 1393–1412 (2003)

[5] Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning, pp. 148–156 (1996)

[6] Garner, S.R.: Weka: The waikato environment for knowledge analysis. In: Proc. of the New Zealand Computer Science Research Students Conference, pp. 57–64 (1995)

[7] Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)

[8] Kleinberg, E.M.: Stochastic discrimination. Annals of Mathematics and Artificial Intelligence 1 (1990)

[9] Kleinberg, E.M.: On the algorithmic implementation of stochastic discrimination. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(5), 473–490 (2000)

[10] Kleinberg, E.M., Ho, T.K.: Pattern recognition by stochastic modeling. In: Proceedings of the Third International Workshop on Frontiers in Handwriting Recognition, pp. 175–183. Partners Press (1993)

[11] Prior, M., Windeatt, T.: Over-fitting in ensembles of neural network classifiers within ecoc frameworks. In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS 2005. LNCS, vol. 3541, pp. 286–295. Springer, Heidelberg (2005)

[12] Prior, M., Windeatt, T.: Parameter tuning using the out-of-bootstrap generalisation error estimate for stochastic discrimination and random forests. In: International Conference on Pattern Recognition, pp. 498–501 (2006)

An Information Theoretic Perspective on Multiple Classifier Systems

Gavin Brown

School of Computer Science, University of Manchester,
Kilburn Building, Oxford Road, Manchester, M13 9PL
gbrown@cs.man.ac.uk
http://www.cs.man.ac.uk/~gbrown/

Abstract. This paper examines the benefits that *information theory* can bring to the study of multiple classifier systems. We discuss relationships between the mutual information and the classification error of a predictor. We proceed to discuss how this concerns ensemble systems, by showing a natural *expansion* of the ensemble mutual information into "accuracy" and "diversity" components. This natural *derivation* of a diversity term is an alternative to previous attempts to *artificially define* a term. The main finding is that diversity in fact exists at multiple orders of correlation, and pairwise diversity can capture only the low order components.

1 Introduction

Information Theory sparked a revolution in the practice of electronic communications [1] and has since been successfully applied in countless fields, from anthropology to biology to cosmology. In the last decade or so, it has found significant uptake in Machine Learning. Suppose there is a message Y, encoded and sent to us by a friend through a communications channel, that we receive as a signal X. We would like to decode the received signal X, and recover the correct message Y; that is, we will perform a decoding operation, $\hat{Y} = g(X)$. In Machine Learning terms, we imagine that the friend transmitting the message has access to a particular object, for which Y is the correct class label. They 'encode' the object as a feature vector X. Our task is to decode that feature vector and recover the correct class label, using our predictor function $g(\cdot)$. Using this analogy, information theory provides us with a language and a set of mathematical tools to analyze the situation. One of the most interesting observations it can provide is a bound on the error of our predictor, dependent on the chosen features X. This bound, known as *Fano's inequality*, applies for *any* predictor: be it a simple decision stump, or a nonlinear support vector machine.

We can also use information theory to understand multiple classifier systems. To make the link, consider the received signal X not to be a set of features, but as a set of classifier outputs, which we will use to form an ensemble. In this case, the predictor $g(\cdot)$ corresponds to the ensemble combiner function. In this work we investigate the link in detail, in particular addressing the notion of ensemble diversity.

This paper is structured as follows. Section 2 provides a tutorial introduction to the basics of information theory, including the lesser known concept of *multivariate* mutual information. Section 3 describes how an understanding for the concept of diversity can *naturally* emerge as an expansion of the ensemble mutual information. Section 4 uses this result to characterize and explain the behaviors of Adaboost versus Bagging, sections 5 and 6 present related work and conclude with a look ahead to what advantages this approach might bring to MCS.

2 Background

In this section we review the required elements of information theory, and their relation to Machine Learning. Due to space limitations this is necessarily brief; for an extended treatment the reader might consult reference [2] or [3].

2.1 Information Theory Basics

The fundamental unit of information theory is the *entropy* of a random variable [1]. The entropy, denoted $H(X)$, quantifies the uncertainty present in the distribution of X. It is defined[1] as,

$$H(X) = -\sum_{i=1}^{|X|} p(x_i) \log p(x_i). \tag{1}$$

The base of the logarithm is arbitrary, but decides the "units" of the entropy. When using base 2, the units are 'bits', when using base e, the units are 'nats'. To compute this, we need an estimate of the distribution $p(X)$. This is estimated by frequency counts from data, that is $p(x_i) = \frac{\#x_i}{N}$, the fraction of observations taking on value x_i from the total number of observations N.

If the distribution is highly biased toward one particular event $x \in X$, i.e. little uncertainty over the outcome, then the entropy is low. If all events are equally likely, i.e. maximum uncertainty over the outcome, then $H(X)$ is maximal[2]. Following the rules of standard probability theory, entropy can also be *conditioned* on other events. The *conditional entropy* of X given Y is denoted,

$$H(X|Y) = -\sum_{j=1}^{|Y|} p(y_j) \sum_{i=1}^{|X|} p(x_i|y_j) \log p(x_i|y_j). \tag{2}$$

This can be thought of as the amount of uncertainty remaining in X after we learn the outcome of Y.

[1] In this work we restrict ourselves to discrete RVs, and note $z \log(z) \to 0$ with $z \to 0$.
[2] In general, $0 \leq H(X) \leq \log(|X|)$.

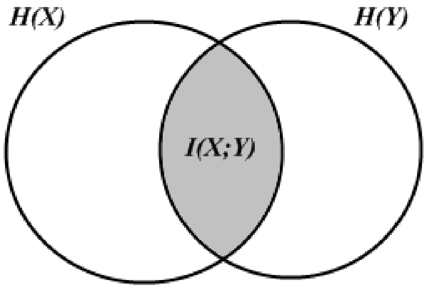


Fig. 1. Illustration of various information theoretic quantities

We can now define the *Mutual Information* between X and Y, i.e. the amount of information *shared* by X and Y, as follows.

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= \sum_X \sum_Y p(xy) \log \frac{p(xy)}{p(x)p(y)}. \end{aligned} \tag{3}$$

It should be noted that Mutual Information is symmetric, i.e. $I(X;Y) = I(Y;X)$. The relation between all these quantities can be seen in figure 1. The Mutual Information can also be conditioned on other events—the *conditional mutual information* is,

$$\begin{aligned} I(X_1;X_2|Y) &= H(X_1|Y) - H(X_1|X_2Y) \\ &= \sum_Y p(y) \sum_{X_1} \sum_{X_2} p(x_1x_2|y) \log \frac{p(x_1x_2|y)}{p(x_1|y)p(x_2|y)}. \end{aligned} \tag{4}$$

This can be thought of as the information still shared between X_1 and X_2 after the value of Y is revealed. The conditional mutual information will emerge as a particularly important property in understanding the message of this paper.

2.2 Relationship to Machine Learning

Suppose there is a message Y, that was sent through a communications channel, and we received the value X. We would like to decode the received value X, and recover the correct Y. That is, we will perform a decoding operation, $\hat{Y} = g(X)$. In ML terms: Y is the original (unknown) class label distribution, X is the particular set of features chosen to represent the problem, and g is our predictor. The set of features chosen may or may not be sufficient to perfectly recover Y; that is, there may be an error in prediction. Information theory can provide a bound on $p(\hat{Y} \neq Y)$, for *any* predictor g.

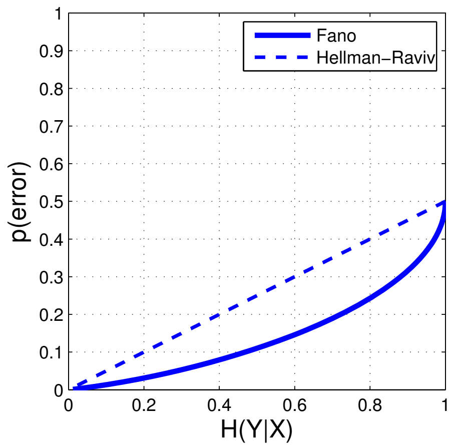


Fig. 2. Fano's inequality [4] provides a lower bound on the Bayes rate, while Hellman-Raviv [5] provides the upper bound. Picking features to reduce conditional entropy (equivalent to maximising mutual information) causes this bound to be minimized.

The error of predicting target variable Y from input X is tightly bounded by two inequalities [4,5]. The bounds state,

$$\frac{H(Y) - I(X;Y) - 1}{\log(|Y|)} \leq p(g(X) \neq Y) \leq \frac{1}{2}H(Y|X). \tag{5}$$

In order to maximise the chances of our predictor guessing the correct class label, we should have maximum $I(X;Y)$. Given the definition (3), this is equivalent to *minimizing* $H(Y|X)$, illustrated in figure 2. As the mutual information $I(X;Y)$ grows, the bound is minimized—whether or not the bound can be reached depends on the ability of our classifier, i.e. the function $g(X)$.

For example, if the conditional entropy is measured to be $H(Y|X) = 0.4$, then the minimum error rate by *any* classifier lies in the range $[0.079, 0.2]$. In other words, no classifier can possibly achieve better than error 0.079 with features X, and there exists a classifier that can achieve at least error 0.2. It should be noted that, in real ML problems, since we only ever have access to a *sample* of X (not the full distribution) this is in practice an estimated bound on the *training* error. We will investigate relations to the the generalization error in section 4.

We have now covered the basic properties of information theory. To complete the background necessary for this paper, we now briefly review the lesser-known topic of *multi-variate* mutual information.

2.3 Multi-variate Mutual Information

While Shannon's mutual information $I(X;Y)$ measures dependence between a *pair* of variables, the multivariate form, known as *Interaction Information* [6], can account for dependencies among *multiple* variables. For a set of size 2, the Interaction Information reduces to Shannon's definition. For *three* random variables, the Interaction Information is

$$I(\{X_1, X_2, X_3\}) = I(X_1; X_2|X_3) - I(X_1; X_2), \tag{6}$$

that is, a difference of the conditional mutual information and the simple mutual information. The case for n variables is defined recursively. A full treatment of this advanced topic is not possible given the limited space; for more information the reader is referred to reference [7]. The interaction information turns out to be useful in understanding the nature of ensemble diversity, which we will explore in the following section.

3 Mutual Information and Ensemble Classifiers

One of the long-standing problems in the MCS literature is to understand the nature of ensemble *diversity*. We know that ensemble members should exhibit some level of accuracy. We also know that ensemble members should not be identical, exhibiting some level of diversity. However, quantifying these statements has proved challenging [8]. In this section we take an information theoretic perspective.

3.1 Why Is Diversity So Elusive?

To answer this question [9] we return to one of the most well-known results in the MCS literature concerning the diversity issue. Tumer & Ghosh [10] related the ensemble classification error to the correlations between the individual predictor outputs. They showed that the error of a *linearly* combined ensemble could be decomposed neatly into accuracy and diversity components. This exemplary early work sparked much effort to find the corresponding accuracy-diversity terms for a majority voting ensemble. A fundamental message of this work is that *we should not expect the majority vote ensemble error to similarly decompose into additive accuracy-diversity terms.*

The neat situation in [10] is due to the linearity of the combination operator, and bias-variance properties of the squared loss function. When we have a *nonlinear* combination operator, and a *zero-one* loss function, the situation is more complicated. It is well appreciated that there exists no unique definition of bias and variance for zero-one loss. In the same fashion, there is no unique definition of *covariance* (diversity) with this loss function; instead, the literature has spawned a myriad of diversity definitions [8] with desirable and undesirable properties.

It is often the case in Machine Learning to use a *surrogate* loss function, and minimise that instead of the actual one of interest. Adaboost is the prime example of this—the distribution updates in the algorithm *do not directly minimise* classification error, but instead minimize a surrogate, an exponential loss which *bounds* the classification loss. In this way, when the exponential loss is small, we can be guaranteed the classification loss will also be at least as small. In the following section we take a similar approach, remembering that the classification error rate can be bounded by the *mutual information*, using Fano's inequality.

3.2 A 'Natural' Definition of Diversity

In this section we show a diversity term emerges naturally when we measure the ensemble mutual information. This draws on a recent result in the feature selection literature [7], described and adapted for the MCS community in the Appendix. For a set of classifiers $S = \{X_1, ..., X_M\}$, remembering that our objective is to maximise $I(X_{1:M}; Y)$, we have the expansion,

$$I(X_{1:M}; Y) = \sum_{i=1}^{M} I(X_i; Y) - \sum_{\substack{\boldsymbol{X} \subseteq S \\ |\boldsymbol{X}|=2..M}} I(\{\boldsymbol{X}\}) + \sum_{\substack{\boldsymbol{X} \subseteq S \\ |\boldsymbol{X}|=2..M}} I(\{\boldsymbol{X}\}|Y). \qquad (7)$$

The expansion consists of three terms. The first, $\sum_{i=1}^{M} I(X_i; Y)$ is the sum of each individual classifier's mutual information with the target. Since the mutual information is actually only a *bound* on the accuracy, not the *actual* accuracy, it is misleading to say this is an 'accuracy' term. Instead, we refer to the first term as the *relevancy* of a classifier output to the target. The final combination function g will determine if this provides good accuracy in combination with the other classifiers.

The second contains terms of the form $I(\{\boldsymbol{X}\})$ and is independent of the class label Y, and so is the closest analogy to the (now almost mythical) concept of 'diversity'. It measures the interaction information among *all possible subsets of classifiers*, drawn from the ensemble. We refer to this as the ensemble *redundancy.* Notice this term is *subtractive* from the overall mutual information. A large value of $I(\{\boldsymbol{X}\})$ indicates strong correlations between the classifiers, and reduces the value of $I(X_{1:M}; Y)$, and hence the overall achievable accuracy.

The third contains terms of the form $I(\{\boldsymbol{X}\}|Y)$ and is a function of the class label Y. This therefore does not correspond to the folklore definition of 'diversity', that it should be a function solely of the classifier outputs. We call this the *conditional redundancy.* Notice that this term is *additive* to the ensemble mutual information. While it is commonly accepted that we should have low correlations between ensemble members, this term indicates that we in fact need *strong class-conditional correlations.* The balance between these conditional and unconditional terms is similar to aiming for a small within-class variance (maximizing the dependency $I(\{\boldsymbol{X}\}|Y)$) and a large between-class variance (minimizing the dependency $I(\{\boldsymbol{X}\})$).

3.3 Low-Order and High-Order Diversity

We have found that through an expansion of the ensemble mutual information, terms which we might call 'diversity' appear naturally. The *redundancy* is a traditional diversity term, and the *conditional redundancy* is the same form but conditioned on the class label. The sum of these two values is what we refer to as the "*diversity*" of the classifier set. It should be noted that the summations in eq(7) are *over all possible subsets of classifiers drawn from the ensemble.* We can expand this sum over subsets, to give us a breakdown of diversity,

$$\begin{aligned} I(X_{1:M};Y) = \sum_{i=1}^{M} I(X_i;Y) &- \sum_{|\boldsymbol{X}|=2} I(\{\boldsymbol{X}\}) + \sum_{|\boldsymbol{X}|=2} I(\{\boldsymbol{X}\}|Y) \\ &- \sum_{|\boldsymbol{X}|=3} I(\{\boldsymbol{X}\}) + \sum_{|\boldsymbol{X}|=3} I(\{\boldsymbol{X}\}|Y) \\ &- \dots \qquad\qquad + \dots \\ &- \sum_{|\boldsymbol{X}|=M} I(\{\boldsymbol{X}\}) + \sum_{|\boldsymbol{X}|=M} I(\{\boldsymbol{X}\}|Y). \end{aligned}$$

This breakdown has the form,

$$\begin{aligned} I(X_{1:M};Y) = \textit{Individual Mutual Info } &+ \textit{2-way diversity (pairwise)} \\ &+ \textit{3-way diversity} \\ &+ \textit{...-way diversity} \\ &+ \textit{M-way diversity} \end{aligned}$$

where the diversity measure is the multivariate mutual information. This expansion reflects the true complexity of the accuracy-diversity issue. Diversity is *not* simply a pairwise measure between classifiers, such as the Q-statistics or the Double-Fault measures. Diversity in fact exists on numerous *levels* of interaction between the classifiers.

4 Monitoring Low-Order Diversity Components

In the previous section we showed that diversity exists at multiple levels of correlation within an ensemble. If the classifiers were statistically independent, then all diversity terms would be zero, and we would have simply $I(X_{1:M};Y) = \sum_{i=1}^{M} I(X_i;Y)$. If the classifiers only exhibited pairwise interactions, the breakdown be as above but omitting the 3-way and above diversity terms. This assumption of pairwise interactions gives us,

$$I(X_{1:M};Y) \approx \sum_{i=1}^{M} I(X_i;Y) - \sum_{j=1}^{M} \sum_{k=j+1}^{M} I(X_j, X_k) + \sum_{j=1}^{M} \sum_{k=j+1}^{M} I(X_j, X_k|Y) \quad (8)$$

The ensemble information is thus approximated by a sum of the relevancy, the pairwise redundancy, and the pairwise conditional redundancy. In figures 3, 4, and 5 we monitor these three components to characterize the behavior of Adaboost and Bagging. All information measurements are made on *training* data, and used to explain the performance on test data. Examining the pairwise components we find Adaboost succeeds by decreasing redundancy, but has no effect on the conditional term. Bagging has no effect on either, reflected in the poor test error. Further comment is provided in the figure captions.

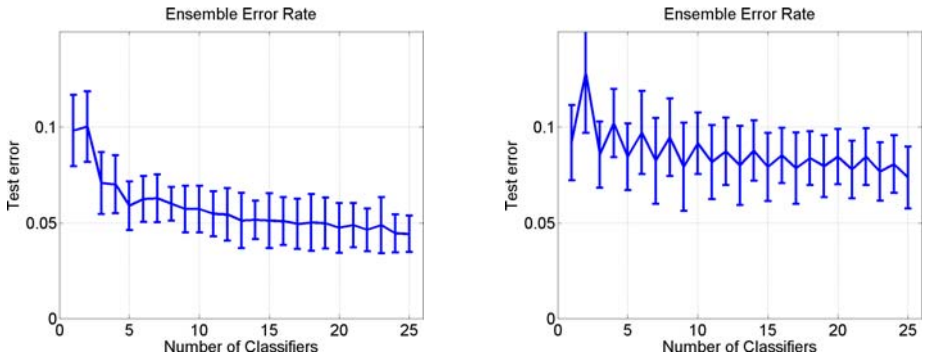


Fig. 3. Adaboost (left) and Bagging (right) errors using decision stumps on the Breast Cancer data. Graphs show standard deviation over ten trials of 2-fold cross validation.

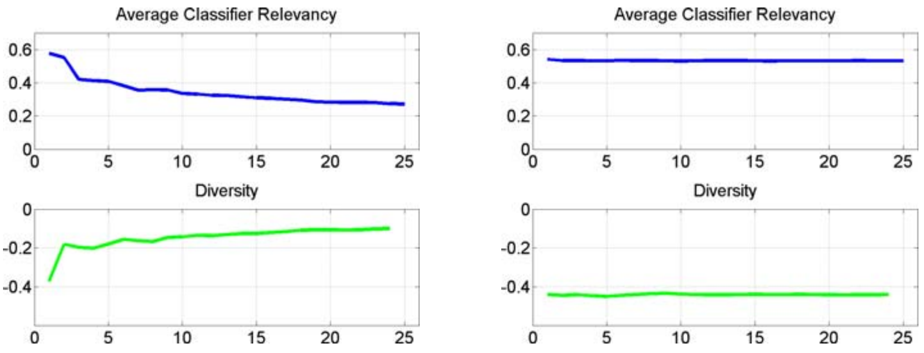


Fig. 4. The Relevancy-Diversity tradeoff. On the left we see the average relevancy of Adaboost classifiers decreases over time, but the diversity component compensates this by also rising. On the right, Bagging maintains almost constant classifier relevancy and very low diversity, explaining the poor test error in figure 3.

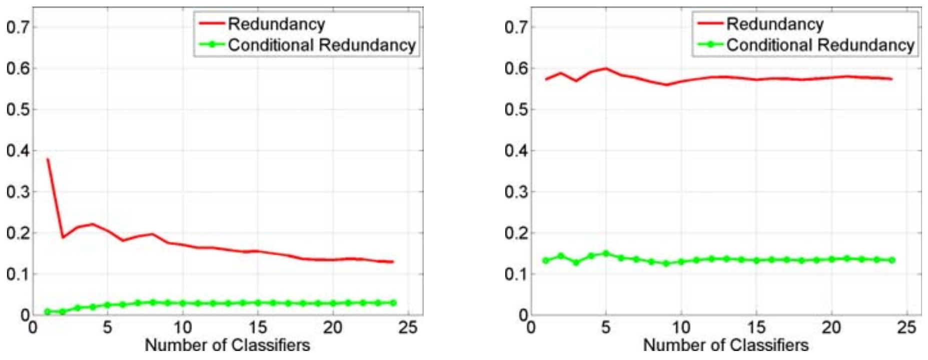


Fig. 5. Second order components of the ensemble mutual information. Adaboost (left) decreases the redundancy of its classifiers, though maintains constant conditional redundancy. Bagging (right) allows the redundancy to rise very slightly at small ensemble size, but has no significant effect on either component.

5 Related Work

Meynet and Thiran [11] suggest a heuristic cost function, designed to balance ensemble accuracy with diversity. The cost function consists of two information theoretic terms. The first is simply the average mutual information between each ensemble member and the class label, which they call the Information Theoretic Accuracy, $ITA = \frac{1}{M}\sum_{i=1}^{M} I(X_i;Y)$. The second is the reciprocal of the average pairwise mutual information between ensemble members, which they call the Information Theoretic Diversity,

$$ITD = \Big(\frac{1}{\binom{M}{2}}\sum_{j=1}^{M}\sum_{k=j+1}^{M} I(X_j;X_k)\Big)^{-1}. \tag{9}$$

Thus, the task is to simultaneously maximise ITA and ITD, though it is clear that a tradeoff will occur between the two. The authors represent the tradeoff by a second-order polynomial: the Information Theoretic Score is defined,

$$ITS = (1 + ITA)^3.(1 + ITD) \tag{10}$$

Comparing the form of ITA and ITD to the results in section 3.2, it is clear that ITS includes two of the necessary components to take account of pairwise interactions between ensemble members. The final term necessary is the class-conditional $I(X_i;X_j|Y)$, and the higher-order terms are assumed zero. The main difference between this heuristic and the current work is that ITS was *hand-designed*, whereas we have shown a natural *derivation* of a diversity term.

6 Conclusion

This paper examined the issue of ensemble diversity from an information theoretic perspective. A major advantage of information theoretic criteria is they capture higher order statistics of the data. In contrast, the squared error criterion can capture only second-order statistics. The main finding was an expansion of the ensemble mutual information which naturally involves “accuracy” and “diversity” component, although diversity is shown to exist at several levels, having low and high order elements.

The advantage of this approach is that $g(\cdot)$ can be *any* function, that is, any ensemble combiner function. In this paper we showed preliminary results with the majority vote combiner, as this has traditionally been of most interest regarding the ‘diversity’ question. Extensions to this work might assess how effective different combiner functions are at ‘decoding’ the information contained in the ensemble.

References

1. Shannon, C.: A mathematical theory of communication. Bell Syst. Tech. J. 27(3), 379–423 (1948)
2. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-Interscience, New York (1991)
3. MacKay, D.: Information Theory, Inference and Learning Algorithms. Cambridge University Press, Cambridge (2003)
4. Fano, R.: Transmission of Information: Statistical Theory of Communications. Wiley, New York (1961)
5. Hellman, M., Raviv, J.: Probability of error, equivocation, and the Chernoff bound. IEEE Transactions on Information Theory 16(4), 368–372 (1970)
6. McGill, W.: Multivariate information transmission. IEEE Trans. Inf. Theory 4(4), 93–111 (1954)
7. Brown, G.: A New Perspective on Information Theoretic Feature Selection. In: Proceedings of Intl. Conf. on Artificial Intelligence and Statistics (2009)
8. Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2006)
9. Kuncheva, L.: That Elusive Diversity in Classifier Ensembles. In: Proc. 1st Iberian Conf. on Pattern Recognition and Image Analysis, pp. 1126–1138 (2003)
10. Tumer, K., Ghosh, J.: Error Correlation and Error Reduction in Ensemble Classifiers. Connection Science 8(3-4), 385–403 (1996)
11. Meynet, J., Thiran, J.: Information Theoretic Combination of Classifiers with Application to AdaBoost. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 171–179. Springer, Heidelberg (2007)

Appendix: Expansion of the Ensemble Mutual Information

Theorem 1
Given a set of classifiers $S = \{X_1, ..., X_M\}$, and a class label Y, their Shannon mutual information can be expanded as

$$I(X_{1:M}; Y) = \sum_{T \subseteq S} I(\{T \cup Y\}), \qquad |T| \geq 1. \tag{11}$$

That is, the Shannon Mutual Information between $X_{1:M}$ and Y expands into a sum of Interaction Information terms. Note that $\sum_{T \subseteq S}$ should be read, "sum over all possible subsets T drawn from S".

Proof: See ref [7].

Example: As an illustrative example for an ensemble of size $M = 3$, the Shannon information between the joint variable $X_{1:3}$ and a target Y can be re-written as

$$\begin{aligned} I(X_{1:3}; Y) = {} & I(\{X_1, Y\}) + I(\{X_2, Y\}) + I(\{X_3, Y\}) \\ & + I(\{X_1, X_2, Y\}) + I(\{X_1, X_3, Y\}) + I(\{X_2, X_3, Y\}) \\ & + I(\{X_1, X_2, X_3, Y\}). \end{aligned} \tag{12}$$

Each term can then be separated into class unconditional $I(\{\boldsymbol{X}\})$ and conditional $I(\{\boldsymbol{X}\}|Y)$ according to the standard definition of interaction information. This gives us the expansion found in the main body of this paper.

Constraints in Weighted Averaging

Amber Tomas

The University of Oxford, Department of Statistics
1 South Parks Road, Oxford OX2 3TG, United Kingdom

Abstract. Weighted averaging of classifier outputs is used in many MCSs, yet is still not well understood. Several empirical studies have investigated the effect that non-negativity and sum-one constraints have on the error rate of weighted averaging rules, but there is little theory available to understand the results.

In this paper we study how constraints on the weights affect the location of the decision boundary of a MCS using weighted averaging. This allows us to explain many of the empirical findings, and suggest guidelines for when the application of constraints may or may not be appropriate. We also consider how these results relate to the analytical framework first proposed by Tumer and Ghosh [5].

1 Introduction

Suppose we have a K-class classification problem and M classifiers whose outputs we combine according to the *weighted averaging* combining rule

$$\hat{p}(k|\boldsymbol{x}) = \sum_{i=1}^{M} w_i \hat{p}_i(k|\boldsymbol{x}),\ \ k = 1, 2, \ldots, K, \tag{1}$$

where $\hat{p}_i(k|\boldsymbol{x})$ is the estimate output by the ith component classifier of the probability that an observation $\boldsymbol{x}$ was generated from the class with label k. Commonly the special case of *simple averaging* is used, in which case all the weights w_i are equal to $1/M$. Given an input $\boldsymbol{x}$, we assume a classification $\hat{c}(\boldsymbol{x})$ is made according to the rule

$$\hat{c}(\boldsymbol{x}) = \text{argmin}_j \sum_{l=1}^{K} L(j,l)\hat{p}(l|\boldsymbol{x}), \tag{2}$$

where $L(j,l)$ is a loss function which specifies the loss incurred when classifying an observation from class l as having come from class j, for $l, j = 1, \ldots, K$.

Given that we use this rule, a method must be determined for obtaining the value of the weights. This typically proceeds in two stages: firstly it is decided which constraints (if any) should be applied to the weights, and then a value for the weights is obtained by some method such that they satisfy the chosen constraints. In this paper we consider the effect of two types of constraint:

1. Sum-Wconstraint: $\sum_{i=1}^{M} w_i = W$,
2. Non-negativity constraint: $w_i \geq 0$ for all $i = 1, \ldots, M$.

The most common sum-Wconstraint is sum-1, for the simple reason that this is a necessary condition for the combined outputs $\hat{p}(k|\boldsymbol{x}),\ k = 1, \ldots, M$ to sum to one. If the outputs are to be interpreted as probabilities, then it is necessary to apply both the sum-1 and non-negativity constraints.

Both Breiman [1] and LeBlanc and Tibshirani [4] explored empirically the accuracy of weighted averaging, where the weights were selected to minimise some sum of squares criteria. In each case, it was found that some constraint on the weights was necessary for good performance, and that the non-negativity constraint produced the best results. LeBlanc and Tibshirani [4] also considered a slightly different combining rule to that of equation (1) which allowed different weights for each class. In this case they found that the sum-1 constraint alone did not produce results as good as when using the non-negativity constraint. Further, Breiman [1] found that when forcing the non-negativity constraint, additionally constraining the weights to sum to one made little difference, and in fact the sum of the weights tended to be close to one anyway. These findings differ somewhat from those of Ting and Witten [6]. Similarly to LeBlanc and Tibshirani [4] they allowed different weights for each class, but they found that enforcing the non-negativity constraint made little difference to the accuracy of the classifier. However, they commented that enforcing non-negativity avoided the need to interpret negative weights.

Although such empirical guidelines are useful, there is still little theoretical understanding of the conditions under which enforcing the non-negativity constraint is likely to improve upon the accuracy of a single classifier. In section 2 we present some theoretical results based on analysis of the location of the decision boundary which help to explain the behaviour noted empirically. In section 3 we show how our findings relate to the framework for analysing classifier decision boundaries proposed by Tumer and Ghosh [5] and extended to weighted averaging by Fumera and Roli [2].

2 Effect of the Constraints on the Location of the Decision Boundary

The analysis in this section is based on analysing the location of the decision boundary of the weighted average classifier (2). Under this rule, the decision boundary between two classes labelled j and j' is the set of points

$$\left\{\boldsymbol{x} : \sum_{l=1}^{K} L(j,l)\hat{p}(l|\boldsymbol{x}) = \sum_{l=1}^{K} L(j',l)\hat{p}(l|\boldsymbol{x}),\right.$$
$$\left.\sum_{l=1}^{K} L(j,l)\hat{p}(l|\boldsymbol{x}) > \sum_{l=1}^{K} L(k,l)\hat{p}(l|\boldsymbol{x})\ \forall\ k \neq j, j'\right\},$$

which is a subset of the set

$$\left\{\boldsymbol{x} : \sum_{l=1}^{K} L(j,l) \sum_{i=1}^{M} w_i \hat{p}_i(l|\boldsymbol{x}) = \sum_{l=1}^{K} L(j',l) \sum_{i=1}^{M} w_i \hat{p}_i(l|\boldsymbol{x}),\right\}. \tag{3}$$

2.1 Non-negativity Constraint

Theorem 1. *Under classification rule (2), when constraining the weights to be non-negative the decision boundary of the final classifier can not lie in the interior of any region of the feature space where all classifiers output the same label.*

Proof. Denote the interior of the region in which the ith classifier will classify an observation as class j by $\mathcal{R}_j^i$, i.e.

$$\mathcal{R}_j^i = \text{int}\left(\left\{\boldsymbol{x} : j = \text{argmin}_c \sum_{l=1}^{K} L(c,l)\hat{p}_i(l|\boldsymbol{x})\right\}\right).$$

Now let $\mathcal{R}_j^*$ denote the interior of the region in which all classifiers will classify an observation as class j, i.e. $\mathcal{R}_j^* = \bigcap_i \mathcal{R}_j^i$. Then for all i, for $\boldsymbol{x} \in \mathcal{R}_j^*$,

$$\sum_{l=1}^{K} L(j,l)\hat{p}_i(l|\boldsymbol{x}) < \sum_{l=1}^{K} L(j',l)\hat{p}_i(l|\boldsymbol{x}).$$

Hence for $w_i \geq 0,\ i = 1, 2, \ldots, M$, for $\boldsymbol{x} \in \mathcal{R}_j^*$

$$\sum_{i=1}^{M} w_i \sum_{l=1}^{K} L(j,l)\hat{p}_i(l|\boldsymbol{x}) < \sum_{i=1}^{M} w_i \sum_{l=1}^{K} L(j',l)\hat{p}_i(l|\boldsymbol{x}).$$

Hence there does not exist an $\boldsymbol{x} \in \mathcal{R}_j^*$ such that

$$\sum_{i=1}^{M} w_i \sum_{l=1}^{K} L(j,l)\hat{p}_i(l|\boldsymbol{x}) = \sum_{i=1}^{M} w_i \sum_{l=1}^{K} L(j',l)\hat{p}_i(l|\boldsymbol{x}).$$

Therefore, from (3) we can see that no points in the region $\mathcal{R}_j^*$ are on the decision boundary between classes j and j'. A similar argument holds to show that no point in $\mathcal{R}_{j'}^*$ is on the decision boundary between classes j and j'. Hence the decision boundary of the combined classifier can not be in a region where all the classifiers output the same label.

The significance of this result is that it gives a necessary condition for optimal performance of the weighted averaging rule with non-negativity constraints, expressed in terms of the decision boundaries of the classifiers which are combined: if the optimal decision boundary is not within the region where the classifiers output different labels, i.e. "disagree", then the Bayes (optimal) error can not be attained for any value of the weights[1]. However, even when it is the case that the optimal decision boundary is within the "region of disagreement" it may not be possible to attain the Bayes error for any value of the weights w_i, depending

[1] As an aside, this is also a necessary (and sufficient) condition for there to exist a classifier selection mechanism with error equal to the Bayes error.

on how flexible the decision boundary of the classifier is and the complexity of the optimal boundary.

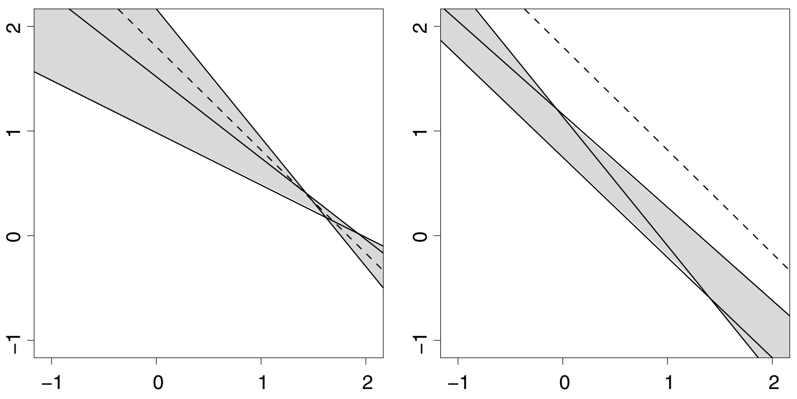
Given that the optimal boundary is within the region of disagreement, making this region tighter about the Bayes boundary can lower, but not increase, the maximum possible error rate of the classifier. Applying a non-negativity constraint can provide regularisation by reducing the space of possible decision boundaries. In addition, this result shows that if using a non-negativity constraint we should use component classifiers whose decision boundaries lie close to the optimal boundary, but are spread about it in an even way. Alternatively, one could say that if we expect to have such a set of classifiers then, due to the added regularisation, it is likely that using a non-negativity constraint will result in a lower error rate than when using unconstrained estimates.

If the region of disagreement of the classifiers does not contain the Bayes boundary, then it is possible that using a weighted averaging rule with non-negativity constraint will result in a relatively high error rate, depending on how far away from this region the optimal boundary lies. In such a scenario it is likely that the classifiers whose decision boundaries are closest to the optimal boundary should have the largest weights, in which case the error rate is likely to be lower than that of simple averaging. However, we may be able to obtain an even lower error rate by allowing negative weights, in which case the bound of theorem 1 does not hold.

For example, consider a toy classification problem consisting of two normally distributed populations. The mean of class 1 is $(0, 0)$ and that of class 2 is $(1, 1)$. The variance covariance matrix of each class is equal to the identity matrix, and the prior probability an observation is from class 1 is 0.7. The Bayes error for this example is 0.2041. In figure 1(a) the component classifiers have been trained on independent training sets of size 80 drawn from the population. The optimal boundary is within the region of disagreement (shaded). In figure 1(b) the classifiers were trained on independent training sets of size 80 which consisted of an equal number of observations from class 1 and class 2, and so are biased towards class 2. In this case the region of disagreement does not contain the optimal boundary.

The error rates of the classifier corresponding to using the best non-negative weights, the best unconstrained weights and simple averaging are shown in table 1. This shows that the benefit of using unconstrained weights for the example shown in figure 1(a) is small compared to the benefit for the example shown in figure 1(b). In the latter case, using simple averaging corresponds to an added error 1.5 times that of the best non-negative weights, yet almost 38 times larger than when using the best unconstrained weights.

In practice it is likely that the small increase in added error of using simple averaging over weighted averaging in the case of figure 1(a) is outweighed by the error introduced by weight estimation [3]. However, in a case such as figure 1(b) the potential benefit of using unconstrained weights compared to simple averaging is unlikely to be outweighed by the loss of accuracy due to weight estimation. Therefore, a non-negativity constraint should only be applied if the method used to obtain the classifiers to be combined is unlikely to be biased.



(a) "Unbiased classifiers" (b) "Biased classifiers"

Fig. 1. The decision boundaries of the combined classifiers are shown as solid lines whilst the optimal boundary is dashed. The region of disagreement of the classifiers is shaded.

Table 1. Optimal error rates for the examples shown in figures 1(a) and 1(b) under different constraints on the weights. The Bayes error is 0.2041.

	example (a)		example (b)	
constraint	best weights	error	best weights	error
non-negativity	$(0.26, 0.70, 0.04)$	0.2045	$(0.00, 1.00, 0.00)$	0.2264
unconstrained	$(0.26, 0.70, 0.04)$	0.2045	$(-1.65, 1.98, 0.67)$	0.2050
simple averaging	$(0.33, 0.33, 0.33)$	0.2074	$(0.33, 0.33, 0.33)$	0.2380

2.2 Sum-W Constraint

Theorem 2. *When using weighted averaging, multiplying the weights by a positive constant will not affect the outputs made by the classifier.*

Proof. Suppose we multiply all the weights by a constant $c > 0$. Then rather than output $\hat{p}(k|\boldsymbol{x})$, the result of weighted averaging will be $c\,\hat{p}(k|\boldsymbol{x})$, for $k = 1, \ldots, K$. Clearly

$$\text{argmin}_j \sum_{l=1}^{K} L(j,l) c\,\hat{p}(l|\boldsymbol{x}) = \text{argmin}_j \sum_{l=1}^{K} L(j,l)\hat{p}(l|\boldsymbol{x}),$$

so the labels output by the classification rule (2) will not be changed.

Theorem 3. *When using weighted averaging, multiplying the weights by a negative constant will result in changes in the label output by the classifier for all inputs $\boldsymbol{x}$.*

Proof. If we multiply the weights by a constant $c' < 0$, then as above the result will be that the outputs of weighted averaging will also be multiplied by $c' < 0$.

One can see that

$$\begin{aligned}\text{argmin}_j \sum_{l=1}^{K} L(j,l)c'\hat{p}(l|\boldsymbol{x}) &= \text{argmax}_j \sum_{l=1}^{K} L(j,l)\hat{p}(l|\boldsymbol{x}),\\ &\neq \text{argmin}_j \sum_{l=1}^{K} L(j,l)\hat{p}(l|\boldsymbol{x}),\end{aligned}$$

so using rule ([2]) will give different outputs to the case that we multiply the weights by a positive constant c.

The first result shows that the optimal weights are not unique, as we could always multiply them by a positive constant and obtain the same classifier outputs. Therefore, applying a sum-Wconstraint makes it more likely that there exists a unique optimal vector of weights. However, if the optimal weights are such that their sum is negative, say, then theorem [3] implies that there does not exist a set of optimal weights with positive sum. Therefore, constraining the weights to sum to one, for example, will in some cases exclude the optimal solution, and possibly many good solutions as well. However, the sum-Wconstraint is obviously looser than the non-negativity constraint, which constrains the sign of all the weights as well as (indirectly) the sign of their sum. For illustration, suppose we combine three classifiers and an optimal value for the unconstrained weights is $(-4, 1, 1)$. If we normalise these weights to sum to one then we must multiply by -0.5, resulting in weights $(2, -0.5, -0.5)$. Because we have multiplied by a negative constant, theorem [3] shows that the labels output by the classifier have changed and so are no longer optimal. In this example, one can see that if we were to constrain the weights to sum to a positive constant, we would exclude the optimal unconstrained solution.

These results imply that applying a sum-1 constraint in addition to a non-negativity constraint is unlikely to result in significant reductions in error rate, though does convey some regularisation by limiting the space of optimal weight vectors. Because the sum-1 constraint is not as strict as the non-negativity constraint, if the optimal unconstrained weights are all positive then one would expect to do better when enforcing the non-negativity constraint than the sum-1 constraint. Applying a sum-1 constraint without the non-negativity constraint could have a detrimental effect on classification performance compared to the unconstrained case if the sum of the optimal weights is negative, and otherwise may result in slightly improved weight estimates due to the relatively small amount of regularisation. Note that the sum of the optimal weights is only likely to be negative if there is at least one individual classifier with a relatively high error rate. This is because, in a two-class classification problem, theorem [3] implies that multiplying the weights by a negative constant will swap the labels output by the classifier. Therefore, using an individual weight $w_i < 0$ is equivalent to including a classifier which always outputs the opposite labels to the ith classifier. Note that this also allows for some interpretation of negative weights: a negative weight indicates that the errors made by the corresponding classifier are as "useful" to the MCS as the correct classifications made by those classifiers with positive weight.

3 Relation to Previous Work

The results of section 2 are useful in helping to explain the circumstances in which non-negativity and sum-Wconstraints are appropriate. Previous study of the decision boundary by Fumera and Roli [2] in the framework of Tumer and Ghosh [5] has assumed that the weights are non-negative and sum to one. We now show that if we consider the possibility of unconstrained weights, that similar conclusions to those above can be drawn from this framework. In addition, because this framework allows one to model the exact location of the decision boundary and not just bounds on its location, we can gain a better understanding of the non-negativity constraint.

The framework applies most easily to the case of a one-dimensional feature space, so for simplicity we only consider this case. In addition, we assume there are only two classes (labelled l and k), that the 0–1 loss function is used and that the decision boundary consists of only one point. In what follows we will denote the optimal boundary by x^*, the decision boundary of the weighted averaging classifier by x_{WA} and the decision boundary of the ith classifier in the combination by x_i. Then following Tumer and Ghosh [5], consider the outputs of the component classifiers as being composed of the true conditional class probability plus an error term, i.e.

$$\hat{p}_i(k|x) = p(k|x) + \epsilon_i(k|x).$$

If we linearise $p(k|x)$ and $p(l|x)$ about the Bayes boundary x^*, i.e.

$$p(k|x) \approx p(k|x^*) + (x - x^*)p'(k|x^*), \tag{4}$$

and assume that $\epsilon_i(k|x) = \epsilon_i(k)$ for all x close to x^*, then the decision boundary x_{WA} can be expressed as

$$\begin{aligned} x_{\mathrm{WA}} &= \{x : \hat{p}(k|x) = \hat{p}(l|x)\} \\ &\approx \frac{1}{sW}\sum_{i=1}^{M} w_i[\epsilon_i(l) - \epsilon_i(k)] + x^*, \end{aligned} \tag{5}$$

where $s = p'(k|x^*) - p'(l|x^*)$ and $W = \sum_i w_i$. Setting $w_i/W = 1$ in equation (5) gives $x_i \approx \frac{1}{s}[\epsilon_i(l) - \epsilon_i(k)] + x^*$. Hence if we denote the distance between x_{WA} and x^* by d_{WA}, then

$$d_{\mathrm{WA}} \approx \sum_{i=1}^{M} \frac{w_i}{W}(x_i - x^*), \text{ i.e.}$$

$$d_{\mathrm{WA}} \approx \sum_{i=1}^{M} \frac{w_i}{W} d_i, \tag{6}$$

where d_i denotes the distance between the boundary of the ith classifier and x^*. This shows that under the assumptions of this framework, the decision boundary of the weighted averaging classifier is offset from the optimal boundary by the weighted average of the offsets of the individual classifiers, with weights w_i/W.

Tumer and Ghosh [5] showed that under the assumptions given earlier, we can express the added error of a classifier with decision boundary x_b (i.e. the error rate of the classifier minus the Bayes error) as

$$\mathrm{AE} \approx \frac{s}{2}(x^* - x_b)^2.$$

Hence, from (6), we can express the added error of the weighted average as

$$\begin{aligned} AE_{\mathrm{WA}} &\approx \frac{s}{2}(d_{\mathrm{WA}})^2, \\ &\approx \frac{s}{2}\left(\sum_{i=1}^{M} \frac{w_i}{W} d_i\right)^2 . \end{aligned} \tag{7}$$

That is, the added error of the weighted average is proportional to the square of the weighted average of the offsets of the individual classifiers.

Optimising the weights is then a matter of minimising the added error (7). To allow comparison with the results of section 2.1, suppose we constrain the weights to be non-negative. If all the classifiers have offset d_i of the same sign, i.e. their decision boundaries all lie on the same side of x^*, then expression (7) implies that we can minimise AE_{WA} by using only the best component classifier, i.e. that with the smallest absolute offset and hence lowest error. That is, if x^* does not lie in the region of disagreement, then the framework shows that we should select the classifier whose decision boundary is closest[2] to the optimal boundary. Furthermore, (7) shows that it is only possible to obtain zero added error if there is at least one component classifier with negative offset, and one with positive offset, i.e. if the region of disagreement contains the optimal boundary. Also note that expression (6) implies that multiplying the weights by a positive constant will not affect the location of the decision boundary. In fact, because we have only two classes, in this case it can be seen that multiplying the weights by a negative constant will not change the location of the decision boundary either, though in this case the labels output by the classifier will be reversed.

Hence this analysis yields the same conclusions about the effect of the non-negativity constraint as that of section 2.1. In addition, by modelling the location of the decision boundary we can say something about the value of the optimal weights. Firstly, note that from (6) we can see that even with two constraints on the weights, if $M > 2$ then the optimal weights will not be unique [3]. So suppose for simplicity that we have only two classifiers, and decide to enforce the non-negativity constraint. Then it makes sense to also apply the sum-1 constraint

[2] By "closest" to the optimal boundary we mean with respect to the probability density.

[3] In practice it is likely that due to the complexity of the optimal decision boundary this is unlikely to occur.

in order to ensure uniqueness of the optimal weights and interpretation of those weights as probabilities. If d_1 and d_2 are of the same sign, then the added error (7) will be minimised if we select the best of the two classifiers. If the d_i are of opposite signs, then the added error will be zero if we set $w_i^{-1} \propto |d_i|$. Now suppose that we do not apply the non-negativity constraint. In this case we can minimise the added error (7) by setting $w_i^{-1} \propto I_i|d_i|$, where $I_1, I_2 = 1$ if d_i are of opposite signs, and $I_1 = 1, I_2 = -1$ if the d_i are of the same sign. In this case it is preferable not to enforce a sum-Wconstraint because a sum-Wconstraint with $W > 0$ is only appropriate if $|d_1| > |d_2|$, and otherwise we should use a sum-Wconstraint with $W < 0$. Because in practice we wont know the signs of the d_i it is not possible to know in advance the appropriate sign of W.

From (7), $\mathrm{AE}_i \propto d_i^2$, so $|d_i| \propto \sqrt{\mathrm{AE}_i}$. Therefore, in certain circumstances as described above the error rate of the weighted average classifier is minimised by setting $w_i^{-1} \propto \sqrt{\mathrm{AE}_i}$. Whilst we can not estimate the d_i directly, we can estimate the added errors AE_i. This result is slightly different to that of Fumera and Roli [2] who showed that if the classifiers are expected to be unbiased and the weights constrained to be non-negative, then the weight of the ith classifier should be inversely proportional to its expected added error, i.e. $w_i^{-1} \propto E[\mathrm{AE}_i]$. However, this does not contradict our findings because we are concerned with the scenario that the weights are determined after the classifiers have been trained rather than beforehand.

4 Conclusions and Discussion

In this paper we explored the effect that constraining the weights can have on the location of the decision boundary of the weighted average classifier. We showed that enforcing a non-negativity constraint results in regularisation due to constraining the region in which the decision boundary can lie. This result implies that the non-negativity constraint is likely to improve performance of the weighted averaging rule provided that the classifiers which are combined are not all systematically different to the optimal boundary. If the optimal boundary lies outside the region where the combined classifiers disagree, then enforcing this constraint will potentially result in a classifier with larger error rate than when using unconstrained weights.

We also showed that under certain conditions applying a sum-1 constraint may improve performance, as it limits the number of optimal solutions. However, if the optimal weights sum to a negative constant, then enforcing a sum-1 constraint will be detrimental to performance of the classifier. Therefore, it would seem sensible to enforce a sum-1 constraint only when a non-negativity constraint is also used.

These findings explain the empirical results discussed in section 1. Furthermore, it is hoped that the theoretical analysis will inform sensible choice of constraints for new or unusual classification problems, or for new methods of training the classifiers which are combined. Because the effect of the constraints considered depends on the relation of the component classifiers to the optimal classifier, which we do not know, it may be more fruitful in future to focus on other forms of regularisation, such as shrinkage.

References

1. Breiman, L.: Stacked Regressions. Machine Learning 24, 49–64 (1996)
2. Fumera, G., Roli, F.: Performance Analysis and Comparison of Linear Combiners for Classifier Fusion. In: Caelli, T.M., Amin, A., Duin, R.P.W., Kamel, M.S., de Ridder, D. (eds.) SPR 2002 and SSPR 2002. LNCS, vol. 2396, pp. 424–432. Springer, Heidelberg (2002)
3. Fumera, G., Roli, F.: A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems. IEEE Trans. Pattern Anal. Mach. Intell. 27(6), 942–956 (2005)
4. Le Blanc, M., Tibshirani, R.: Combining Estimates in Regression and Classification. Journal of the American Statistical Association 91, 1641–1650 (1996)
5. Tumer, K., Ghosh, J.: Analysis of Decision Boundaries in Linearly Combined Neural Classifiers. Pattern Recognition 29, 341–348 (1996)
6. Ting, K.M., Witten, I.H.: Issues in Stacked Generalization. Journal of Artificial Intelligence Research 10, 271–289 (1999)

FaSS: Ensembles for Stable Learners

Kai Ming Ting[1], Jonathan R. Wells[1], Swee Chuan Tan[1],
Shyh Wei Teng[1], and Geoffrey I. Webb[2]

[1] Gippsland School of Information Technology,
[2] Clayton School of Information Technology,
Monash University, Australia
{kaiming.ting,jonathan.wells,james.tan,
shyh.wei.teng,geoff.webb}@infotech.monash.edu.au

Abstract. This paper introduces a new ensemble approach, Feature-Space Subdivision (FaSS), which builds local models instead of global models. FaSS is a generic ensemble approach that can use either stable or unstable models as its base models. In contrast, existing ensemble approaches which employ randomisation can only use unstable models. Our analysis shows that the new approach reduces the execution time to generate a model in an ensemble with an increased level of localisation in FaSS. Our empirical evaluation shows that FaSS performs significantly better than boosting in terms of predictive accuracy, when a stable learner SVM is used as the base learner. The speed up achieved by FaSS makes SVM ensembles a reality that would otherwise infeasible for large data sets, and FaSS SVM performs better than Boosting J48 and Random Forests when SVM is the preferred base learner.

Keywords: Local models, stable learners.

1 Introduction

Existing ensemble methods such as Bagging, Random Forests, Random Subspace and Boosting generate multiple **global models** from a single learning algorithm through randomisation (or perturbation) in order to improve predictive accuracy relative to a single model. The reliance on global models and randomisation (or perturbation) also means that only unstable base learners can be used for these ensemble methods because only unstable base learners generate sufficient global model diversity through randomisation or perturbation. This excludes many stable base learners that, when applied directly, may produce more accurate individual models for the given learning task. Unstable learners will generate substantially different models when there is a small perturbation on the training data; whereas stable learners generate models that differ little in the context of small perturbations. Examples of stable learners are Naive Bayes, k-nearest neighbour classifiers and support vector machines. Decision tree learners are a typical example of unstable learners.

This paper introduces a fundamentally different approach to ensemble learning, which induces diverse **local models**, rather than global models, in distinct local regions of the feature-space. The proposed ensemble approach constructs all partitions of the feature-space that are defined over a fixed number of attributes and learns a local model in each partition. This approach can be applied to both stable and unstable learners, and trains faster than existing ensemble methods because each local model is learned using a substantially smaller data set than learning a global model using the whole data set. The reduction in training time will be more pronounced when the learning algorithm has high order polynomial time complexity. In addition, our approach will continue to get improvement by increasing the level of localisation for as long as the data quantity is sufficient to support the level of localisation.

The proposed work is distinguished from existing work on ensemble methods based on randomisation (such as Bagging [2], Random Forests [3]) or perturbation (Boosting [11]), feature subset selection/search for ensembles [8,9,5], and localised modelling (such as NBTree [6], Lazy Bayesian Rules [14]) by

- Employing a complete enumeration of feature-space subdivisions to build a set of local model variants for each point in the feature-space; thus eliminating the need to use randomisation or perturbation to build model variants;
- Utilising local models in a way that provides more model diversity than existing learners which either heuristically identify or randomly select a small set of feature subsets to build global models (see Section 2.3 for more details);
- Enabling either randomised or enumerated implementation.

We show in this paper that the proposed approach improves the predictive accuracy of the ensemble and decreases the time required to generate an individual model, as the level of localisation increases. For example, in one of the largest data sets we used, building a SVM ensemble of 455 models takes less than one-hundredth of the time required to train one single SVM model! The approach works for both stable and unstable models such as decision trees, k-nearest neighbours and support vector machines (SVM). In this paper, we focus on the stable learner SVM and show that the proposed approach performs significantly better than Boosting in terms of predictive accuracy, and is significantly faster if training a single SVM requires a long time.

2 Feature-Space Subdivision

A **local model** formed from instances similar to one we wish to classify will often perform better than a **global model** formed from all instances [4]. However, in the general case we do not know the relevant distance metric so do not know what local neighborhood to use. We propose to use many local neighborhoods, creating an ensemble by applying the base learner in each.

Our approach is to subdivide the feature-space into non-overlapping local regions in a single subdivision; and ensure that different subdivisions provide the distinct local neighbourhoods for each point in the feature-space. There are many ways a feature-space can be subdivided. Instead of using heuristics, we subdivide the feature-space exhaustively based on a user-specified number of features to control the level of localisation. The set of exhaustive feature-space subdivisions forms the basis to develop a new ensemble method which aggregates all local models or a random subset generated in the set. We call the proposed method Feature-Space Subdivision or FaSS.

We describe the enumerated version of FaSS in Section 2.1 and provide the time complexity analysis in Section 2.2, and explain the randomised version of FaSS in Section 2.3.

2.1 Definition and Implementation

Let X be the input n-dimensional feature space with A_i denote the i-th feature, and Y denote the set of classes representing the underlying concept, which is to be learnt from a training set of size m, $D := \{(x_i, y_i) | i \in \{1, 2, \ldots, m\}\}$.
Let $e : X \rightarrow \{0, 1\}$ be a Boolean-valued function which consists of a conjunction of h atoms over $A_1, \ldots, A_n$. The subspace and its complement confined by e are given as follows: $r := \{x \in X \mid e(x) = 1\}$, $\bar{r} := X \backslash r$.

We define an h-subdivision as: $R = \bigcup_i r_i = X$, where $\forall\ i \neq j,\ \ r_i \cap r_j = \emptyset$.

r_i are mutually exclusive subspaces that cover the entire feature space using conjunctions of h features, and $h \leq \frac{n}{2}$. A learning algorithm f is used to build a local model $f(r_i)$ for each subspace r_i.

Let r_i^j be a subspace i of an h-subdivision j. The set of all distinct h-subdivisions, in which no two r subspaces are equivalent or a subset of another between any two h-subdivisions, is defined as follows:

$$Rh := \{R \mid \forall\ j \neq k, r_.^j \not\subseteq r_.^k\}, \tag{1}$$

$$|Rh| = C_h^n, \tag{2}$$

where C_h^n is the binomial coefficient.

FaSS is an ensemble method which combines all local models generated from Rh or a subset of Rh.

An example of FaSS in binary domains. In a domain of n binary attributes, the feature-space can be subdivided into two half-spaces n ways, where each such subdivision uses one of the n attributes. Extending to subdivision using h attributes, the feature-space can be subdivided into $\frac{1}{2^h}$-spaces in C_h^n ways, where h is the number of binary attributes used to do the subdivision. In general, an h-subdivision is one of the possible non-overlapping subdivisions of the feature-space using h attributes. The exhaustive set has C_h^n h-subdivisions for a given feature-space defined by n attributes.

Let us consider that a subspace is a local neighbourhood of a point x in the feature space. Applying the subdivision, each point x has exactly C_h^n different local neighbourhoods; and x is the minimum intersection of all the local neighbourhoods. Each subspace is defined by one conjunction of h conditions. For example, $\{A_1{=}0$ and $A_2{=}1\}$ defines a subspace constrained by attributes A_1 and A_2 and their specified values. The exhaustive set of x's local neighbourhoods contains all the different training data sets from which x can be modelled (by using a learner to produce one local model for each local neighbourhood.)

Models learned from these distinct local neighbourhoods can be viewed as the representative local models for x. For example, if the feature space is defined by four binary attributes A_1, A_2, A_3 and A_4 and x=<A_1=0, A_2=1, A_3=1, A_4=0>; for h=2, six quarter-spaces r_i of two dimensions are shown below, where each column indicates the attributes used (with an implied value for each attribute) in each quarter-space:

$$\begin{array}{cccccc l} A_1 & A_1 & A_1 & A_2 & A_2 & A_3 & \text{Level 1} \\ A_2 & A_3 & A_4 & A_3 & A_4 & A_4 & \text{Level 2} \end{array}$$

which define all six local neighbourhoods for x. This example shows the enumeration method we employed to generate C^n_h h-subdivisions in an FaSS ensemble which can be done without attribute selection.

h specifies the level of localisation—a high h signifies a high level of localisation with a reduced training data size in each subdivision; and only attribute subsets, which do not contain attributes used in r to define a local neighbourhood, are used to build local model for the neighbourhood.

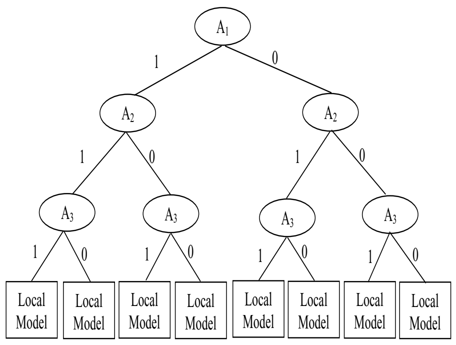
Level Tree. The data structure we propose, which implements the feature-space subdivision mentioned above, is called a *Level Tree*. It is a restricted form of decision tree where each level of the tree must use the same attribute. Fig. 1(a) shows an example of a Level Tree defined by three attributes: A_1, A_2 and A_3. A local model is trained from data in each leaf of the tree and it is attached to the leaf. A Level Tree[1] with local models is thus equivalent to a single global model, ready to predict when a test instance is presented.

Using feature-space subdivision, the structures of all possible h-subdivision Level Trees (without local models) can be generated with minimal cost by enumeration since no attribute selection is required, unlike in the case of ordinary decision trees. Subdivision for numeric attributes will be considered in the same manner as a cut-point selection in an ordinary decision tree using a heuristic such as information gain. As a result, though each attribute can appear at one level only in a Level Tree, the cut-points used for a numeric attribute on different nodes of the same level can be different.

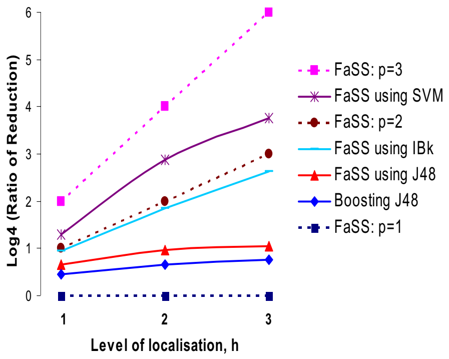
The training data is filtered through the branches in the Level Tree as each node is created. If the training data run out before a branch reaches the required level h, a leaf is formed and the majority class of its parent is used for prediction. Otherwise, a local model is trained for each branch that has a level h.

The aggregation of all possible Level Trees of h-subdivision forms an ensemble. To make the final prediction, the predictions from all Level Trees in an ensemble are aggregated using a simply majority vote, like in bagging.

[1] As a single model, a Level Tree is similar to an Oblivious Decision Tree (ODT) [7], except that an ODT uses the majority class of the training instances at a leaf to classify whereas a Level Tree uses local models. However, ODT is meant to be used as a single model with no considerations for ensembles.



(a) An example of Level Tree



(b) Execution time reduction

Fig. 1. (a) An example of Level Tree with three levels of localisation using attributes: A_1 at level 1, A_2 at level 2, and A_3 at level 3, and a local model attached to each leaf. Each Level Tree with local models forms a single global model.
Fig. 1. (b) Plot of ratio of reduction (Log_4) at three levels of localisation for FaSS using hypothetical algorithms having time complexities $O(m^3)$, $O(m^2)$ and $O(m^1)$, FaSS using SVM, IBk and J48, and Boosting J48.

2.2 Time Complexity Analysis

Assume an algorithm with time complexity $O(m^p)$ is used to generate 2^h local models in each binary Level Tree of h-subdivisions, where m is the number of training instances and p is a constant. The total execution time of generating a single binary Level Tree is in the order of $2^h m^p$, ignoring the time to generate the structure of a Level Tree which is negligible by comparison in nominal attribute only domains. Further assume uniform data distribution and every increment of h reduces the number of instances in each subdivision by half. Thus, the ratio of reduction in execution time (T_R) for each Level Tree as a result of a single increment of h is given as follows:

$$T_R = \frac{2^h m^p}{2^{h+1}(m/2)^p} = 2^{p-1}$$

Compared to generating a single global model, the ratio of reduction in execution time for generating a single model in FaSS of h-subdivision is thus $2^{h(p-1)}$. Generalising to b-nary trees, where $b \geq 2$, the ratio of reduction will be $b^{h(p-1)}$. This reduction may be viewed as an upper bound because of the uniform data distribution assumption. In contrast, most ensemble methods employing randomisation have the ratio of reduction approximately equal to one since generating a single model and each of its variants from a base learning algorithm requires about the same amount of time when using the same data size.

Fig 1(b) shows an example of the ratios of reduction in execution time of a single model using FaSS with quad trees (i.e., b=4 because each attribute in this domain has four labels) at h=1, h=2, and h=3; and FaSS employs three different base learning algorithms: J48 (decision tree), IBk (k-nearest neighbour where k=5) and SVM which have different time complexities. We also compare them to Boosting using J48 with ensemble sizes equivalent to the three settings of h. The data set (coding) consists of a total of 20000 instances, and the result is averaged over a 10-fold cross-validation. Note that we are applying log_4 to the reduction ratios and include FaSS using three hypothetical algorithms with different p values which show three linear lines with different reduction ratios.

The largest reduction comes from FaSS SVM which is equivalent to FaSS using a hypothetical algorithm with a time complexity between $O(m^2)$ and $O(m^3)$; and the reduction ratios for FaSS IBk is very close to FaSS using a hypothetical algorithm with $O(m^2)$. FaSS J48 appears to behave as FaSS using an algorithm with a time complexity between $O(m)$ and $O(m^2)$. The reduction ratio for Boosting is relatively modest and is less than that for FaSS J48.

2.3 Randomised Version of FaSS

FaSS can be implemented in more than one way, including a randomised version of FaSS which randomises the feature-space subdivision process (i.e., selecting a random subset of Rh), as opposed to the randomisation/perturbation in the instance space like bagging or boosting, and randomisation in the model building process as in Random Forests. This implementation completely avoids the issue of enumerating all C_h^n models in FaSS without randomisation.

Randomised FaSS is different from Random Subspace (RS) [5] in three key aspects. First, RS builds each global model from all data points, albeit with a reduced feature set; whereas FaSS builds local models using local training sets defined by subspaces in R. Second, the diversity of RS models is confined by the set of C_h^n feature subsets only, assuming h is the number of selected features; whereas the model diversity in FaSS is further enhanced by $b^{(h+1)}$ different local models in each of the C_h^n feature subsets, assuming in a b-nary domain. In effect, each Random Subspace model is forced to *exclude* a subset of the features (those not selected) and is free to utilize as it sees fit the remaining features to build a global model, whereas each FaSS model is forced to *include*, albeit implicitly, a subset of features (those selected to build the level tree) and is left free to utilize those remaining to build local models. Third, for an algorithm with $O(nm^p)$, the training time for RS is expected to be reduced by half only, when $h = n/2$ is used in order to have the maximum diversity; whereas the training time for FaSS is expected to be reduced in the order of $(n - h)m^p/b^{h(p-1)}$.

3 Empirical Evaluation

We design our experiment to examine the properties of the proposed FaSS approach and compare it with Bagging and Boosting. The experiment is conducted under the WEKA platform [13]. We use the support vector machine, decision tree and boosting implementations in this platform (i.e., SMO, J48, AdaBoostM1).

The two base learning algorithms are selected because of their differing characteristics that we want to examine in FaSS. They are algorithms which produce different degrees of (un)stable models; have significantly different time complexities. J48 produces an unpruned decision tree whenever it is used; all other settings are as in the default settings. All experiments are carried out on an OpenPOWER 720 Power5 Linux Cluster with 8 gigabytes main memory.

A total of eight plus three large data sets from the UCI Machine Learning Repository [1] are used. All experiments are conducted using 10-fold cross-validation, unless stated otherwise. We measure the performance in terms of error rate and execution time per model. We measure a total of training and testing time in each data set and then divide the total time by the ensemble size to give the average execution time per model in order to demonstrate the time reduction in comparison to that of a single ordinary model.

We first show the performance of FaSS without randomisation in Sections 3.1 and 3.2, and then FaSS with randomisation in Section 3.3.

3.1 FaSS SVM

The results of FaSS SVM are presented in Table 1. In terms of error rate and execution time, FaSS SVM shows that both the error rate and the execution time per model decrease as the level of localisation increases. The geometric means of performance ratios in Table 1 show that FaSS decreases the error rate of SVM by 20% at $h = 1$ and 45% at $h = 3$. There are a few exceptions to the general trend which occur in wave21 and dna. The error rate spikes at high level of localisation in the dna data set are likely to be due to the effect of a low number of training examples relative to the high branching factor and high number of attributes. This data set has 60 nominal attributes, each having four labels, and a data size of 3196 examples only. Many branches in the Level Tree have no or very small number of training examples when $h = 3$. SVM requires more training examples

Table 1. Average error rate and execution time (in seconds) at different ensemble sizes for FaSS SVM (linear kernels); data and ensemble sizes at h=1, 2 and 3

	Average Error				**Time Per Model**					**Ensemble Size**		
		FaSS				**FaSS**			**Data**	**for FaSS at** h		
	SVM	**1**	**2**	**3**	**SVM**	**1**	**2**	**3**	**Size**	**1**	**2**	**3**
coding	28.58	26.10	23.67	20.05	2392	397.45	43.72	13.03	20000	15	105	455
nursery	6.85	3.75	1.57	.35	47.06	7.39	2.79	2.91	12960	8	28	56
dna	7.34	3.77	3.20	7.19	17.02	4.19	2.38	5.14	3186	60	1,770	34,220
wave40	13.72	13.68	13.46	13.42	3.60	1.26	1.02	1.10	5000	40	780	9,880
satimage	13.33	11.84	10.86	9.76	3.51	1.63	1.54	2.19	6435	36	630	7,140
wave21	12.98	13.00	13.16	13.42	2.37	.70	.48	.48	5000	21	210	1,330
segment	6.97	6.41	5.63	4.68	2.04	.70	.57	.52	2310	19	171	969
anneal	12.58	9.91	7.13	6.01	1.89	.53	.32	.30	898	38	703	8,436
Average	12.79	11.06	9.84	9.36	308.69	51.73	6.60	3.21				
Geomean		.80	.65	.55		.27	.14	.14				

in order to produce a reasonable model; thus it can only tolerate up to $h = 2$. The number of poor performing local models in this kind of branches becomes sufficiently large to affect the performance of the ensemble when h is high. Thus, domains with many multi-label attributes require to either have large data size or use a low level of localisation when employing FaSS.

In terms of execution time, the geometric means in Table 1 show that FaSS at $h = 1$ reduces the time to construct and test a model to 27% of SVM at $h = 1$ and 14% at $h = 3$. The general trend persists in every single data set for FaSS SVM from SVM to $h = 1$ and $h = 2$. However, there is a slight increase in execution time at $h = 3$ (relative to $h = 2$) in four of the eight data sets for FaSS using SVM. These are likely to be due to an increase in difficulty (relative to those in the lower levels) in converging to the final SVM local models as a result of small data size in subdivisions at high levels.

3.2 FaSS SVM versus Boosting SVM

Table 2 compares the results between Boosting and FaSS using SVM as the base model. Except in one data set (anneal), Boosting makes little or no improvement on the predictive performance of SVM. In contrast, FaSS reduces error rates significantly in all data sets, except the two waveform data sets. The total time results show that FaSS uses significantly less time than Boosting in the three most time consuming data sets (coding, nursery and dna), in which training a single SVM takes a long time. Notably FaSS of eight SVM models uses only a slightly more time than a single SVM model in nursery, but reduces almost half the error rate. Also note that Boosting SVM takes significantly more time in nursery than a single SVM model because the changed data distribution increases the training time significantly for training individual models.

Table 2. Average error rate and total execution time (in seconds) for SVM, Boosting SVM and FaSS SVM using h=1. Boosting with early stopping (up to 100 iterations) is used here and the average number of boosting iterations over the 10-fold cross validation is also provided. Using SVM as the baseline, the average error results that are better (at 5% level of significance of a pair-wise t-test) are highlighted in boldface.

	Average Error			**Total Time**			**#Iterations**	
	SVM	Boost	FaSS	SVM	Boost	FaSS	Boost	FaSS
coding	28.58	28.58	**26.10**	2392	9797.6	5962	3.9	15
nursery	6.85	6.85	**3.75**	47.1	1570.7	59.2	12.5	8
dna	7.34	7.41	**3.77**	17.0	2295.0	251.2	100.0	60
wave40	13.72	13.72	13.68	3.6	28.0	50.3	3.6	40
satimage	13.33	13.33	**11.84**	3.5	14.3	58.5	3.1	36
wave21	12.98	12.98	13.00	2.4	14.1	14.8	4.2	21
segment	6.97	6.67	**6.41**	2.0	9.2	13.4	6.5	19
anneal	12.58	**8.02**	**9.91**	1.9	18.0	20.1	19.8	38

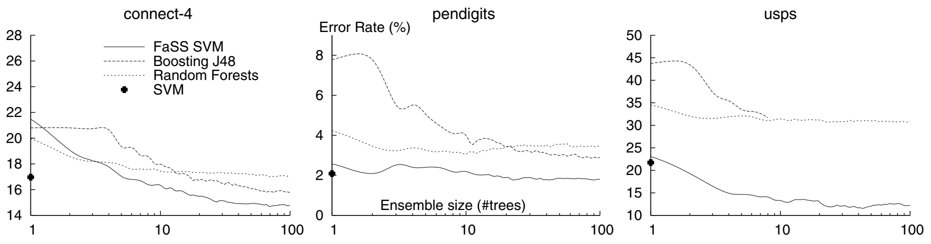


Fig. 2. Error Rate Curves for FaSS SVM, Boosting J48 and Random Forests. This experiment is conducted using a 2:1 split of the given data set for training and testing. Quadratic kernels are used. These data sets have the following characteristics (#attributes, data size): connect-4 (42, 67557); usps (256, 9298); pendigits (16, 10992). FaSS SVM reduces SVM's error rate from 2.1% to 1.8% in pendigits; Boosting stops early at 8 iterations in usps.

3.3 Why Use FaSS Instead of Boosting or Random Forests?

Our results in the previous two sections show that FaSS has two key advantages: (i) the reduction in execution time per model as the level of localisation increases; and (ii) it works well with stable learners. Therefore, FaSS is useful when data size is large, and it is particularly useful in domains where stable learners outperform the alternatives. Figure 2 shows three such examples in which SVM is the most accurate base learner and FaSS (with randomisation) can be used to further improve its predictive performance; and the result shows that FaSS SVM performs significantly better than Boosting J48 and Random Forests.

Bagging and boosting are known to work for unstable models only. Though there are a few works that apply boosting or bagging to SVM, they use either boosting to select a subsample to scale up the efficiency of SVM for large data sets [10] or bagging in conjunction with random subspacing [12] to increase SVM's instability, achieving better performance without the benefit of scaling up the execution time. In contrast, FaSS not only scales up the efficiency of SVM, but also significantly improves its predictive performance, as we have showed in section 3.1. For an additional example in the most time consuming connect-4 data set shown in Figure 2, FaSS SVM of 100 models takes 1.7 days—which is about one-tenth of the time (16.3 days) to train one single SVM model! Boosting SVM of 100 models in this data set is estimated to take about 4.5 years! Boosting or Bagging has no mechanism to reduce the runtime. Boosting or other ensemble methods that build global models would be infeasible for compute time expensive algorithms such as SVM in large data sets.

4 Conclusions

This paper shows the advantages of using local models in ensembles. This is a significant departure from existing ensemble methods that utilises global models through randomisation (or perturbation) on the training data or of the training process. The proposed FaSS approach possesses the same property of existing ensemble approaches, i.e., increasing ensemble size improves an ensemble's predictive performance, with two added advantages.

First, FaSS significantly reduces the execution time for each model with a high level of localisation. Our analysis shows that the ratio of reduction is in the order of $b^{h(p-1)}$ when an algorithm with time complexity $O(m^p)$ is employed to generate local models in FaSS using b-nary Level Trees of h-subdivision. In other words, the reduction becomes more pronounced when the algorithm has high order polynomial time complexity.

Empirical results show that FaSS executes each model faster—FaSS is an effective way to speed up SVM to make ensembles feasible for SVM; and FaSS SVM is significantly faster than Boosting SVM for large data sets.

Second, FaSS is an ensemble approach which is more generic than existing approaches; many of which are limited to unstable models. Both stable and unstable models such as SVM and decision trees can be used in FaSS to improve their predictive performance by increasing the ensemble size through increasing level of localisation. (The result for unstable learners is not shown because of space limitation.) We show that FaSS SVM performs significantly better than Boosting SVM in terms of predictive accuracy, and FaSS SVM performs better than Boosting J48 and Random Forests when SVM is the preferred base learner.

These advantages are a direct result of using local models rather than global models in the ensembles, and the FaSS approach ensures that there are no duplicate local regions in which the same data subset is used to generate multiple local models in the ensemble—ensuring maximum model diversity.

References

1. Asuncion, A., Newman, D.J.: UCI repository of machine learning databases. University of California, Irvine (2007)
2. Breiman, L.: Bagging Predictors. Machine Learning 24, 123–140 (1996)
3. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
4. Frank, E., Hall, M., Pfahringer, B.: Locally Weighted Naive Bayes. In: Proc. of the 19th Conf. on Uncertainty in AI, pp. 249–256 (2003)
5. Ho, T.K.: The Random Subspace Method for Constructing Decision Forests. IEEE Trans. Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
6. Kohavi, R.: Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. In: Proc. of the 2nd KDD, pp. 202–207 (1996)
7. Kohavi, R., Li, C.H.: Oblivious Decision Trees, Graphs, and Top-Down Pruning. In: Proc. of 1995 Intl. Joint Conf. on Artificial Intelligence, pp. 1071–1077 (1995)
8. Opitz, D.: Feature selection for ensembles. In: Proc. of the 16th AAAI, pp. 379–384 (1999)
9. Oza, N.C., Tumer, K.: Input Decimation Ensembles: Decorrelation through Dimensionality Reduction. In: Kittler, J., Roli, F. (eds.) MCS 2001. LNCS, vol. 2096, pp. 238–247. Springer, Heidelberg (2001)
10. Pavlov, D., Mao, J., Dom, B.: Scaling-up support vector machines using the boosting algorithm. In: Proc. of 2000 Intl. Conf. on Pattern Recognition, pp. 219–222 (2000)

11. Schapire, R.E., Singer, S.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37, 297–336 (1999)
12. Tao, D., Tang, X., Li, X., Wu, X.: Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. IEEE Trans. on Pattern Analysis and Machine Intelligence 28(7), 1088–1099 (2006)
13. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. (2005)
14. Zheng, Z., Webb, G.I.: Lazy Learning of Bayesian Rules. Machine Learning 41(1), 53–84 (2000)

Classifying Remote Sensing Data with Support Vector Machines and Imbalanced Training Data

Björn Waske, Jon Atli Benediktsson, and Johannes R. Sveinsson

University if Iceland, Faculty of Electrical and Computer Engineering,
Hajararhagi 2-6, 107 Reykjavik, Iceland
{waske,benedikt,sveinsso}@hi.is
http://www.hi.is

Abstract. The classification of remote sensing data with imbalanced training data is addressed. The classification accuracy of a supervised method is affected by several factors, such as the classifier algorithm, the input data and the available training data. The use of an imbalanced training set, i.e., the number of training samples from one class is much smaller than from other classes, often results in low classification accuracies for the small classes. In the present study support vector machines (SVM) are trained with imbalanced training data. To handle the imbalanced training data, the training data are resampled (i.e., bagging) and a multiple classifier system, with SVM as base classifier, is generated. In addition to the classifier ensemble a single SVM is applied to the data, using the original balanced and the imbalanced training data sets. The results underline that the SVM classification is affected by imbalanced data sets, resulting in dominant lower classification accuracies for classes with fewer training data. Moreover the detailed accuracy assessment demonstrates that the proposed approach significantly improves the class accuracies achieved by a single SVM, which is trained on the whole imbalanced training data set.

Keywords: land cover classification, multispectral, support vector machines, bagging, imbalanced training data.

1 Introduction

Monotemporal applications are often inefficient, when classifying remote sensing data from agricultural areas. Due to great differences in the phenology of planted crops, these regions areas are characterized by great temporal variability of land cover changes. Thus, the classification of multitemporal data sets, containing images that were acquired at different dates during the observation period, appear appropriate in this context and usually improve the classification accuracy. On the other hand, common statistical classifier algorithms seem not adequate in this context and alternative techniques are necessary to handle multitemporal data sets. Support Vector Machines (SVM), which are well known in the field of machine learning and pattern recognition, were introduced in context of remote sensing during the recent years [1]-[3]. In these studies they perform better or at least equal to other classification methods. Another development is the use of multiple classifier systems (MCS) or classifier ensembles [4]. MCS were used successfully in diverse remote sensing application [5] and are particularly interesting for multisource and hyperspectral data sets [5]-[9]. The results demonstrate that classifier systems usually increase the performance of a single classifier and perform at least as good as sophisticated classification strategies as SVM. Bagging [10] is a well known approach for generating classifier ensembles. The concept is based on the random selection of training sample sets. Afterwards an individual classifier is trained on each of these sample sets, resulting in various classifier predictions. The final classification map is generated by combining these individual outputs and often a simple majority vote is used. Whereas in many studies decision tree classifiers are used for constructing an ensemble with bagging, in [11] and [12] SVM were used as a base classifier.

Beside the classifier algorithm, the input data and the available reference data (in context of supervised classifications) have a significant impact on the classification accuracy. Imbalanced training data sets, i.e., the number of training samples from one class is much smaller than from other classes, often results in low classification accuracies for the minor classes. Thus the problem of supervised classifications with imbalanced training data is addressed in several studies [13]-[15]. In [13] for instance, the classification of forest types in remote sensing data is discussed. The authors proposed an SVM strategy, which enables the handling of imbalanced training data. In context of machine learning and pattern recognition classification problems with imbalanced training samples are often handled by (1) over-sampling and (2) under-sampling the training data, respectively [16]. Following the over-sampling strategy, some training patterns of the minor class are resampled before the classifier training, resulting in the same number of samples as the other classes. In the latter one, the size of the major classes is decreased to match the number of samples within the minor class. In the study presented here, a combination of bagging and SVM is used to classify multitemporal remote sensing data set, with imbalanced training data. The approach is applied to a time series of multispectral SPOT data, which was acquired between May and July over an agricultural study site in Western Germany. The paper is organized as follows. In the next section, a short introduction to the conceptual framework of SVM and multiple classifier systems is given. The data sets and preprocessing are described in Section 3. The actual methods for the application of the proposed ensemble strategy are explained in Section 4. Experimental results are presented in Section 5. Section 6 discusses results and conclusion.

2 Classifier Algorithms

2.1 Support Vector Machines

Support vector machines are a binary classifier concept that separates two classes by fitting an optimal separating hyperplane to the training data of two classes in a multi-dimensional feature space. For linearly not separable cases, the input data are mapped into a high dimensional space, using so-called kernel function [17]. A detailed introduction on the general concept of SVM is given in [18] and [19]. A training data set of ℓ samples, in a d-dimensional feature space $\Re^d$, is given by x_i with their corresponding class labels $y_i = \pm 1$, $\Omega = \{(\mathbf{x}_i, y_i) \mid i \in [1, \ell]\}$.

The linear hyperplane $f(x)$ is given by the normal vector w and the bias b, with $|b| / \|w\|$ as the distance between the hyperplane and the origin, where $\|w\|$ is the Euclidean norm from w

$$f_l(x) = wx + b. \tag{1}$$

The margin maximization leads to the following optimization problem:

$$min \left[\frac{w^2}{2} + C \sum_{i=1}^{\ell} \zeta_i \right] \tag{2}$$

where ζ_i denotes the slack variables and C the regularization parameter, which is used to penalize training errors. The SVM decision function for a non-linear separable case is described by:

$$f_n(x) = \left(\sum_{i=1}^{\ell} \alpha_i y_i k(x_i, x_j) + b \right) \tag{3}$$

where α_i are Lagrange multipliers. The kernel function $k(x_i, x_j)$ performs a mapping operation and enables us to work within the newly transformed feature space, only knowing the kernel function. A common kernel function in remote sensing applications is the Gaussian radial basis function (RBF), defined by:

$$k(x_i, x_j) = exp \left[-\gamma \|x_i - x_j\|^2 \right]. \tag{4}$$

The training of an SVM classifier requires the adequate definition of the kernel parameter γ and the regularization parameter C. The constant C is used as a penalty for training samples that located on the wrong side of the hyperplane. It controls the shape of the solution and thus affects the generalization capability of the SVM. However, the use of inadeqaute parameter values might result in a less accurate classification. Often the kernel parameters are determined by a grid-search, using n-fold cross validation. Potential combinations of C and γ are tested in an user defined range and the best combinations for C and γ were selected based on the results of the cross validation.

2.2 Multipe Clasifier Systems

The main idea of a multiple classifier system is to consider several different decisions in the final classification, instead of relying on a single classifier [4]. Ensemble classifier are based on the assumption that a set of independent (i.e., diverse) classifiers produce individual errors, which are not generated by the majority of the other classifiers. Thus, a combination of these classifier outputs can improve the classification accuracy. Consequently, the classifier diversity is a prerequisite and a set of more or less similar classifier (outputs) would not increase the classification accuracy [4]. Classifier ensembles can be generated by combining different classifier methods [6],[9] as well as by a combination of variants of the same algorithm, the so-called base classifier [7],[8]. In [6] for instance, different data sources were classified simultaneous by a neural network and statistical classifier. The different outputs (i.e., class labels) were combined by decision fusion. A decision fusion strategy that is based on SVM was addressed in [8], to combine multispectral remote sensing and SAR data. Each image source was classified separately by an SVM and the original outputs, i.e., the distances from each pixel top the hyperplane, were combined by an additional SVM. The proposed strategy increases the classification accuracy, compared to a single SVM that was applied to the whole data set.

Other multiple classifier systems are based on diverse variants of the same algorithm. Boosting and bagging are perhaps the widest used concepts to generate such a diverse set of classifiers. Breimans bagging [10] uses subsets of training data, also known as bootstrapped aggregates. For each classifier in the ensemble, a random and uniform selection of training samples is performed with replacement, selecting ℓ samples from a training set of same size ℓ. This means that some training samples can be selected several times for a specific training set, whereas other samples are not considered in this particularly bag. Afterwards separate classifiers are trained, using the different training sample sets. The different results are combined for the final prediction, often by a majority vote. The bagging code can be described as follows:

```
Input: A training set $\Omega=\{(\mathbf{x}_j, y_j)\}_{j=1}^{\ell}$, the base classifier $\mathcal{C}_B$ and number
of randomly generated training sets $I$.
 1. FOR $i = 1$ to $I\{$
 2.      $\Omega_i$ = bootstrapped bag from $\Omega$
 3.      $\mathcal{C}_i = \mathcal{C}_B(\Omega_i)\}$
 4. END
 5. the class with the maximum number of votes is chosen
```

Although many classifier ensembles based on decision trees, thanks to the simple handling and fast training times, other studies discussed the use of SVM ensembles, e.g., [11],[12]. In [11] SVM ensembles were constructed by boosting and bagging. In context of remote sensing, SVM ensemble was used for the classification of high-resolution images in [12]. An SVM ensemble was constructed, by training each SVM on a subset of training samples, which was randomly generated without replacement. The experimental results of these studies showed that SVM ensembles can outperform a single SVM in terms of classification accuracy.

3 Study Site and Data Sets

The test site is located near Bonn in the German state North Rhine-Westphalia. The area is dominantly used for agriculture. In this study a multitemporal data set was available, containing multispectral SPOT images (11 May, 24 June, 17 July and 25 July 2006). Thus, the data set comprised information from varying phenological stages. An orthorectification of the imagery was performed, using a digital elevation model and a geocoded reference image. The experiments presented here are focusing on 9 land cover classes: *Arable crops, Cereals, Grassland, Forest, Maize, Orchards, Rapeseed, Root crops, Urban.* A map from a detailed field survey was used for generating the training (100 samples per class) and test sample sets (500 samples per class). Nine different imbalacned training data sets were generated, each time the number of training samples of one land cover class was randomly reduced to 20. Regarding the random generation, each experiment was conducted 5 times, i.e., 5 imbalanced samples sets per class were generated (45 in total).

4 Methods

The imagery was normalized before the classifier training and stretched between 0 and 1. The training of the SVM with Gaussian kernel and the classification were performed using LIBSVM in a MATLAB environment [20]. The kernel parameters of C and γ are determined by a grid-search, using a 3-fold cross validation. Possible combinations of C and γ are tested in an user defined range and the best combinations were used for the final classification. To handle the multiclass problem - SVM were originally developed as binary classifiers - the one-against-one was followed. Each potential SVM was generated, separating one class from another. The final class was determined by a simple majority vote. An usual SVM classification is performed, using the imbalanced training data sets and the original balanced data set, containing 100 samples per class. During the bagging process 20 samples (i.e., the number of sample of the minor class) were selected without replacement from each major class to generate the final balanced training data set. The individual outputs were combined by a simple majority vote to create the final result. For each ensemble 10 individual iterations were performed. As mentioned before, the experiments were conducted five times, using 5 different imbalanced training sets per class. For the final accuracy assessment the results were averaged.

5 Experimental Results

The results demonstrate the impact of imbalanced training data set on the classification accuracy. Whereas the total accuracy is almost not affected, the class accuracies of the minor class are significantly reduced. Using the original balanced data set, with 100 samples per class, the SVM classification achieves an accuracy of 76.7%. This overall accuracy is slightly reduced by 1.5% when imbalanced data sets are used. In contrast to this, the class accuracies are significantly reduced by using imbalanced data sets. However, the effect varies between the different land cover classes: e.g., the accuracies of *Cereals* and *Orchards* are significantly reduced by 44% and 68%, respectively, whereas the effect *Forests* is less sensitive, resulting in a reduction in a reduction of the accuracy by 5.6% (see Fig. 3)

This negative impact of the imbalanced training can be reduced by constructing an SVM ensemble. Compared to the results achieved by a single SVM (with

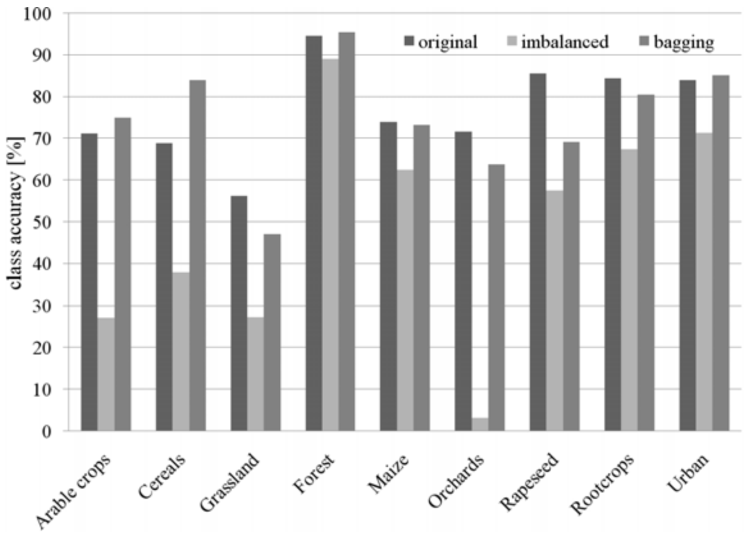


Fig. 1. Class accuracies, using the original and the imbalanced data with and without bagging

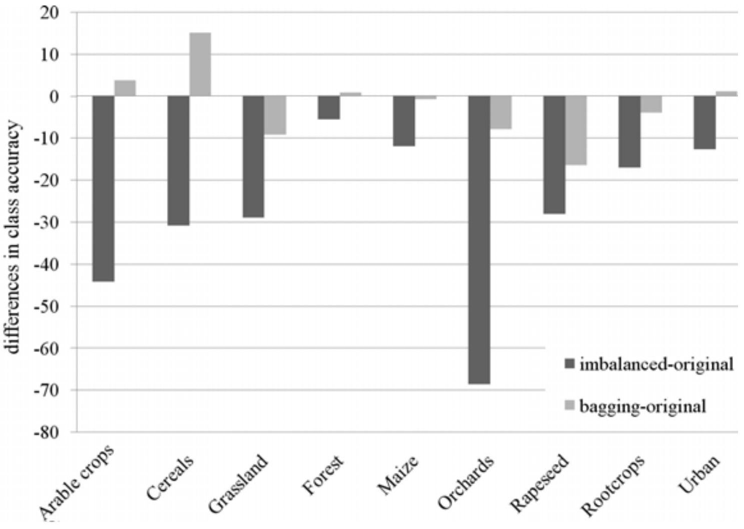


Fig. 2. Differences in class accuracies, using the original and the imbalanced data with and without bagging

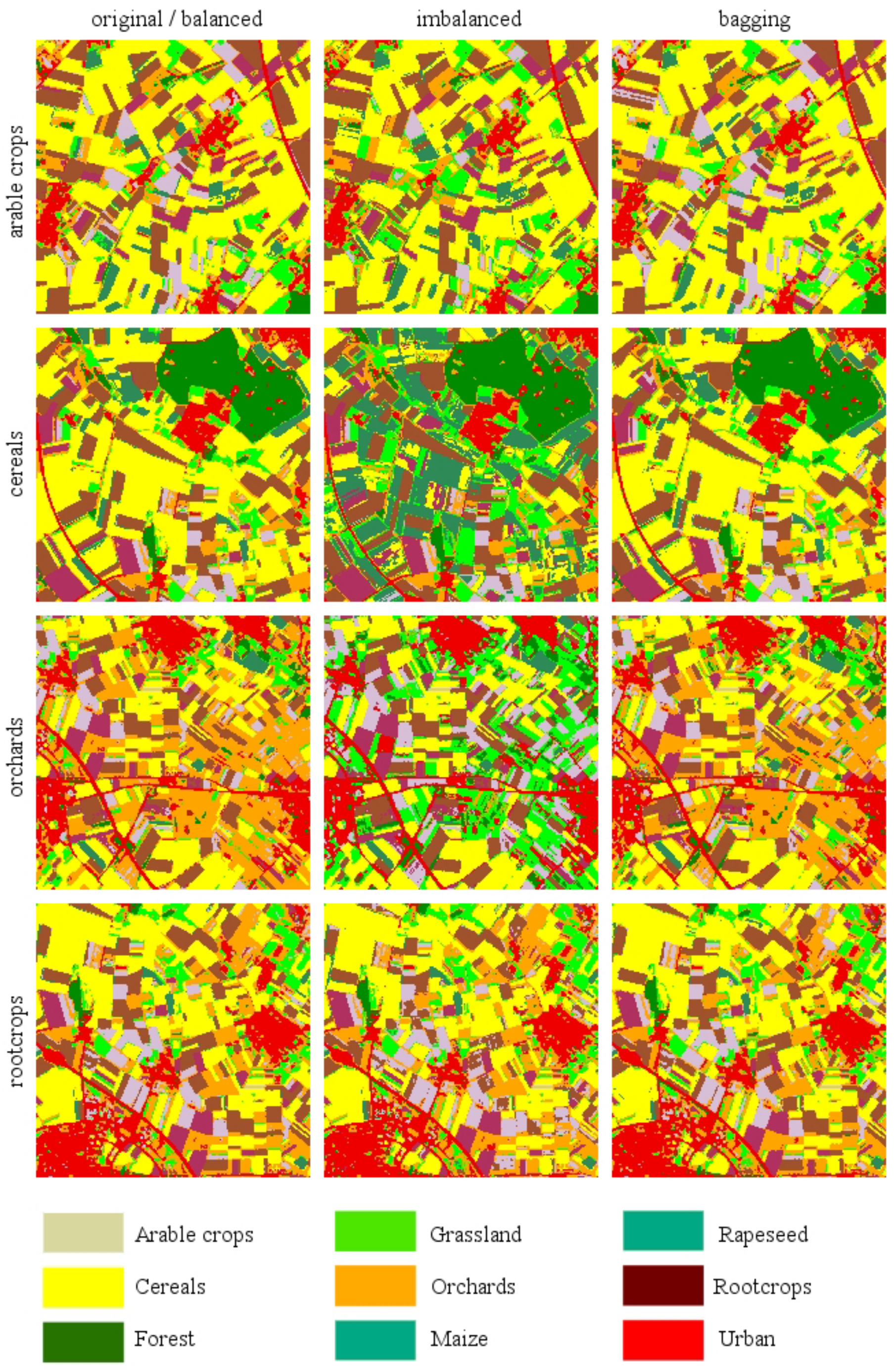


Fig. 3. Differences in class accuracies, using the original and the imbalanced data with and without bagging

imbalanced training data), the accuracy of each minor class is improved by the proposed approach (see Fig. 1). To visualize this effect, the differences in the class accuracies achieved by the three approaches are presented in Fig. 2. Whereas the differences in the accuracies achieved due to imbalanced and the original training data clearly tend towards negative, i.e., the accuracy is significantly reduced, the differences between the accuracies achieved by bagging and the original data are much more balanced.

In addition the impact of the number of iterations within the ensemble was investigated and larger classifier systems were constructed, containing 50 individual classifications (not presented in detail). However, with a few exceptions, the large ensembles could neither improve the total accuracy nor the accuracy of the minor classes.

The visual interpretation of the classification results clearly shows the advantage of the proposed approach (Fig. 3). Comparing the classification maps generated with imbalanced data to the results achieved with the original training set (i.e., balanced with 100 samples per class) it is obvious that many regions belonging the minor classes are neglected. Following the proposed strategy most of these regions are assigned to the minor class, as in the original classification result (see Fig. 3).

6 Discussion and Conclusion

In the presented study, the problem of classifying remote sensing data with imbalanced data sets was discussed and an approach based on multiple classifier systems was proposed. The proposed method is based on the generation of SVM ensembles by resampling the training data (i.e., bagging). The experimental results underline the negative effect of imbalanced training data, resulting in a significant lower classification accuracy of the minor classes. As in other studies, the construction of SVM ensembles with baggin, does not significantly effect the overall accuracy. Nevertheless the proposed method significantly improves the class accuracies of the minor classes. Of course, this requires the generation of additional SVM classifiers, but on the other hand an ensemble with 10 SVM perform very well. Moreover, additional classifiers (i.e., a large ensemble size) do not further improve the class accuracy of the minor classes. It is interesting to underline that the negative effect of imbalanced data as well as the positive effect of the proposed method on the class accuracies varies between different land cover classes. Whereas some classes classified reasonable well with imbalanced training data, the class accuracy of other classes is significantly reduced. One reason might be that some classes can already successfully separate by few training samples, whereas other classes require a higher number of training samples for an adequate classification. Overall the proposed method is well suited for dealing with imbalanced data sets. Due to the general high performance of SVM in context of remote sensing - particularly when classifying multisource and hyperspectral data sets - the proposed strategy seems particularly interesting with respect to real world applications, when dealing with imbalanced data sets on the one hand and with high-dimensional and multisource data sets on the other.

Acknowledgments. This work is funded by the Research Fund of the University of Iceland. The authors would like to thank V. Heinzel, M. Braun and G. Menz from ZFL, Bonn University, for their support within the research project and field works, which were supported by the German Aerospace Center (DLR) and Federal Ministry of Economics and Technology (BMWi) under the project Enviland (FKZ 50EE0404) as well as the European OASIS program (OASIS 58 - CE 6324).

References

1. Huang, C., Davis, L.S., Townshend, J.R.: An assessment of support vector machines for land cover classification. Int. J. Remote Sens. 23, 725–749 (2002)
2. Foody, G.M., Mathur, A.: A Relative Evaluation of Multiclass Image Classification of Support Vector Machines. IEEE Trans. Geosci. and Remote Sens. 42, 1335–1343 (2004)
3. Melgani, F., Bruzzone, L.: Classification of hyperspectral remote sensing images with support vector machines. IEEE Trans. Geosci. and Remote Sens. 42, 1778–1790 (2004)
4. Polikar, R.: Ensemble Based Systems in Decision Making. IEEE Circuits and Systems Magazine 6, 21–45 (2006)
5. Benediktsson, J.A., Chanussot, J., Fauvel, M.: Multiple Classifier Systems in Remote Sensing: From Basics to Recent Developments. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 501–512. Springer, Heidelberg (2007)
6. Benediktsson, J.A., Kanellopoulos, I.: Classification of Multisource and Hyperspectral Data Based on Decision Fusion. IEEE Trans. Geosci. and Remote Sens. 37, 1367–1377 (1999)
7. Briem, G.J., Benediktsson, J.A., Sveinsson, J.R.: Multiple Classifiers Applied to Multisource Remote Sensing Data. IEEE Trans. Geosci. Remote Sens. 40, 2291–2299 (2002)
8. Waske, B., Benediktsson, J.A.: Fusion of Support Vector Machines for Classification of Multisensor Data. IEEE Trans. Geosci. and Remote Sens. 45, 3858–3866 (2007)
9. Waske, B., van der Linden, S.: Classifying multilevel imagery from SAR and optical sensors by decision fusion. IEEE Trans. on Geosci. and Remote Sens. 46, 1457–1466 (2008)
10. Breiman, L.: Bagging predictors. Mach. Learning 24, 123–140 (1996)
11. Kim, H.-C., Pang, S., Je, H.-M., Kim, D., Bang, S.Y.: Constructing support vector machine ensemble. Pattern Recogn. 36, 2757–2767 (2003)
12. Zortea, M., De Martino, M., Serpico, S.: A SVM ensemble approach for spectral-contextual classification of optical high spatial resolution imagery. In: Proc. of IGARSS 2007 Symposium, Barcelona, Spain (2007)
13. Trebar, M., Steele, N.: Application of distributed SVM architectures in classifying forest data cover types. Comp. and Electr. in Agriculture 63, 119–130 (2008)
14. Imam, T., Ting, K.M., Kamruzzaman, J.: z-SVM: An SVM for Improved Classification of Imbalanced Data. In: Sattar, A., Kang, B.-h. (eds.) AI 2006. LNCS, vol. 4304, pp. 264–273. Springer, Heidelberg (2006)

15. Kang, P., Cho, S.: EUS SVMs: Ensemble of Under-Sampled SVMs for Data Imbalance Problems. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4232, pp. 837–846. Springer, Heidelberg (2006)
16. Barandela, R., Valdovinos, R., Sánchez, J.: New Applications of Ensembles of Classifiers. Pattern Analy. & Appl. 6, 245–256 (2003)
17. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
18. Burges, C.J.C.: A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery 2, 121–167 (1998)
19. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2002)
20. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm

Terrain Segmentation with On-Line Mixtures of Experts for Autonomous Robot Navigation

Michael J. Procopio[1], W. Philip Kegelmeyer[1], Greg Grudic[2], and Jane Mulligan[2]

[1] Sandia National Laboratories,
Albuquerque, NM 87185; Livermore, CA 94551
{mjproco,wpk}@sandia.gov
[2] Department of Computer Science,
University of Colorado at Boulder,
Boulder, CO 80309
{grudic,janem}@cs.colorado.edu

Abstract. We describe an on-line machine learning ensemble technique, based on an adaptation of the *mixture of experts* (ME) model, for predicting terrain in autonomous outdoor robot navigation. Binary linear models, trained on-line on images seen by the robot at different points in time, are added to a model library as the robot navigates. To predict terrain in a given image, each model in the library is applied to feature data from that image, and the models' predictions are combined according to a single-layer (flat) ME approach. Although these simple linear models have excellent discrimination in their local area in feature space, they do not generalize well to other types of terrain, and must be applied carefully. We use the distribution of training data as the source of the *a priori* pointwise mixture coefficients that form the soft gating network in the ME model. Single-class Gaussian models are learned during training, then later used to perform density estimation of incoming data points, resulting in pointwise estimates of *model applicability.* The combined output given by ME thus permits models to abstain from making predictions for certain parts of the image. We show that this method outperforms a less sophisticated, non-local baseline method in a statistically significant evaluation using natural datasets taken from the domain.

Keywords: Mixture of Experts, Classifier Ensembles, Local Classifier Accuracy, Online Learning, Terrain Segmentation, Autonomous Robot Navigation.

1 Introduction

Autonomous robot navigation in unstructured outdoor environments is a challenging area of active research and is currently unsolved. The navigation task requires identifying safe, traversable paths that allow the robot to progress towards a goal while avoiding obstacles. Stereo vision allows for obstacle avoidance in the

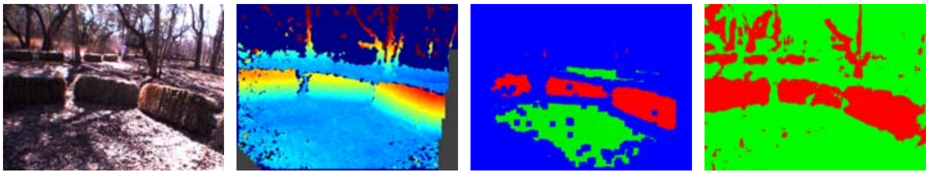


Fig. 1. Demonstration of near-to-far learning using stereo. In (c) and (d), and throughout this paper, red represents nontraversable obstacle (positive); green represents traversable groundplane (negative).

near field (here, within 10 m of the robot). However, navigating solely on near-field terrain readings can lead to a common failure mode in outdoor autonomous navigation where incorrect trajectories are followed due to *nearsightedness*, or an inability to distinguish safe and unsafe terrain in the far field [1].

To address near-sighted navigational errors, *near-to-far learning* is often used [2,3]. The near-to-far approach uses both appearance and stereo information from the near field as inputs for training appearance-based models; these models are then applied in the far field in order to predict safe terrain and obstacles farther out from the robot where stereo readings are unavailable.

Near-to-far learning using stereo is demonstrated in Figure 1. For a given RGB image (1a), stereo disparity is computed using a stereo camera pair; from this data, a groundplane model is fit and subtracted out, resulting in an estimate of groundplane deviation (1b). Near-field stereo labels from both the groundplane and obstacle classes are identified according to small and large groundplane deviation values, respectively (1c); these near-field stereo labels are sampled to create a balanced training set. Next, features are extracted from the image at the pixels of this training set; here, *color histograms* are used [3]. A model is then trained on the resulting near-field feature data. The resulting model is evaluated over the image, including the far field, to arrive at a final terrain predictions (1d). These terrain predictions are used by the robot's path planning system to influence the robot's low-level navigation [1].

Recently, the use of classifier ensembles to learn and store terrain models over time for application to future terrain has been investigated [3,4,5]. These ensembles are constructed dynamically from an on-line *model library* that is maintained as the robot navigates terrain towards some goal. For an incoming image, the outputs of the models in the resulting ensemble are combined, dynamically and in real-time, in a manner designed to optimize predictive performance on far-field terrain. This previous work achieved classifier fusion by taking a linear combination of model outputs using *one* weight per each model in the ensemble. This technique's primary disadvantage is that models cannot be experts *locally*, i.e., at pixel resolution, which is problematic because a given model's discrimination ability may not apply everywhere in the image (i.e., in input space).

2 Terrain Segmentation with Mixtures of Experts

To overcome the shortcomings of the single-weight *non-local* methods noted above, we describe an efficient on-line machine learning ensemble technique. It is based on an adaptation[1] of the *mixture of experts* (ME) model [6,7,8]. The general aim of the approach is to combine multiple experts learned over time by extracting a soft partitioning of the feature space to yield *local accuracy estimates* [9], while also respecting the real-time domain requirement.

On-line mixtures of experts and related mixture models are not novel, nor is the use of Gaussian models to partition the input space and inform where local linear models are applicable. In particular, Sato and Ishii in [10] use the Normalized Gaussian Network [11], or *NGnet*, as the basis for their proposed on-line EM algorithm, used in turn to fit model parameters.

The primary differentiating contribution of this paper is threefold. First, we place explicit emphasis on permitting the mixture of experts to *abstain* from making predictions, where appropriate. Second, our overall approach is for the purpose of binary classification, which requires adaptation and extension of the *NGnets* above. This is contrast to the the approaches taken in [10] and [11], which are framed in a regression context and hence do not model class-conditional data distributions. Finally, this approach has not been previously applied to the terrain segmentation task or to other open problems in the autonomous robot navigation domain.

2.1 Overview of the ME Approach for Terrain Segmentation

During navigation, terrain in an outdoor scene is to be classified as either traversable (groundplane) or nontraversable (obstacle). Two-class appearance-based linear models, trained on-line on images seen by the robot at different points in time, are added to a model library as the robot navigates; the training data is not kept after the model has been added to the library. When terrain segmentation is required, each model in the library is applied to feature data extracted from the current image, and the models' predictions are combined according to the single-layer (flat) ME model. Because mixing coefficients in the ME model are functions of the input data, models can be experts locally in feature space.

Model Abstinence. A key benefit of our approach is model *abstinence*, i.e., when it is determined that a model does not apply to some point $\mathbf{x}$ and hence should be permitted to *abstain* (or mostly abstain) from making a prediction at that point. Not only can individual models abstain, but in our approach, the entire ensemble can abstain at $\mathbf{x}$. Such behavior is desirable in the autonomous

[1] Our approach adopts the mixture of experts architecture from [6]. This is fundamentally a conditional mixture model in which the mixing coefficients, like the expert response, are functions of the input. We do not explicitly fit the model using EM as outlined in [7].

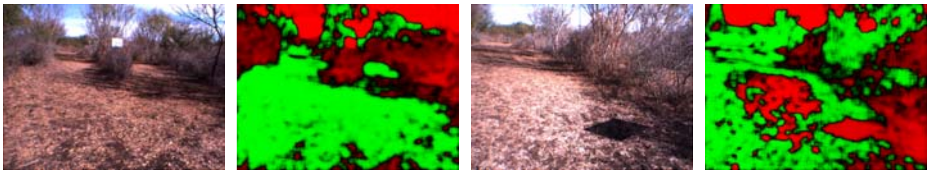

(a) Train $\mathcal{M}_1$ on I_1 (b) Eval $\mathcal{M}_1$ on I_1 (c) Test Image I_2 (d) Eval $\mathcal{M}_1$ on I_2

Fig. 2. Good specialization but poor generalization of linear experts. A model trained on near-field training data from an image (2a) yields reasonable terrain classification on that image (2b). That same model, when applied to a different test image (2c), yields reasonable segmentation in some parts of the image, but not in others (2d).

robot navigation domain, because the robot's path planning system (the *planner*) maintains a probabilistic *cost map* over time used for path planning [1]. If there is not a sufficient basis to make terrain predictions, then the resulting fully abstaining ensemble will result in no updates to the robot's cost map. The next incoming frame would then offer a new opportunity for terrain prediction and cost map updates.

Behavior of Linear Models on Domain Data. Linear models are very efficient to train, which is important in this real-time domain because training and terrain segmentation are done on-line on incoming images while the robot is navigating. We have found that these simple linear models are specialists in their area of feature space, yielding good segmentation on terrain similar to that on which they are trained, but they but do not generalize well to other terrain. Blindly applying these models without regard to their *applicability* to the input is problematic, and can lead to poor terrain prediction as the model is forced to generalize to regions of feature space on which it was not trained.

Fig. 2 shows an example of this.[2] From features extracted from a particular training image (2a), a model is trained. Terrain prediction is reasonable when the model is applied to the image on which it was trained (2b). For a test image with similar terrain appearing later in same data set (2c), the model trained on the original image is applicable only in certain areas (2d), yet still remains equally confident everywhere.

Local Applicability Estimates. For determining where models are applicable, we use the *distribution of training data* as the source of the mixing coefficients that form the soft *gating network* in the ME model. This gating network determines which models are applicable and to what extent for each input (feature

[2] In Fig. 2 and in similar figures throughout the paper, red coloring indicates non-traversable obstacle terrain prediction (model output approaching +1); green coloring indicates traversable groundplane terrain prediction (model output approaching −1); and color intensity indicates prediction confidence, with black representing full uncertainty (model output of 0).

vector) $\mathbf{x}$ in some set of data X. During training of model $\mathcal{M}$, in addition to training the linear model $(\mathbf{w}, b)$, a multivariate Gaussian model $\mathcal{G}$ is fit to each of the C classes of training data (here, $C = 2$). Thus, a model $\mathcal{M}$ comprises $\{\mathcal{G}_{c=1}, \mathcal{G}_{c=2}, \mathbf{w}, b\}$. These Gaussian models are trivial to train and very efficient to evaluate on incoming feature data.

Later, during evaluation (i.e., when segmenting terrain for an incoming image), the density models $\mathcal{G}$ are used to determine how similar the incoming data is to the data on which a previously learned expert was trained; the density model thus provides pointwise estimates of model applicability. The mixing coefficients, which are functions of $\mathcal{G}$ and $\mathbf{x}$, are combined with the expert output at $\mathbf{x}$ according to the ME model to yield the final ensemble output.

Because they are learned on the same training data, a natural concern would be that the mixing coefficients (model applicability estimates) and the model output are not independent. We examined this and determined that the two variables are poorly correlated ($R^2 \approx 0.4$), and conclude that they measure different things.

This approach is similar in principle to that proposed by Grudic *et al.* [12], who also sought a mechanism for applying models only where applicable. There, as here, density models are used to inform when and where to apply models. In that approach, histogram-based density models are used, trained on decision boundary distances of holdout data evaluated through the linear model. Our approach, in contrast, is based on Gaussian density models learned directly from training data. Further, in Grudic's approach, the final classifier output is the output of the density model response, the input for which is the output of the experts (i.e., decision boundary distances). In contrast, in our approach, final classifier output is a fusion of the density model response with the expert output, per the ME model.

Mixture of Experts Model. The mixture of experts (ME) model [6,7,8] on which our approach is based is a type of *conditional mixture model* where the mixing coefficients are functions of the input, shown in Eq. 1:

$$p(\mathbf{t}|\mathbf{x}) = \sum_{k=1}^{K} \pi_k(\mathbf{x}) p_k(\mathbf{t}|\mathbf{x}) \ , \tag{1}$$

where the individual component densities $p_k(\mathbf{t}|\mathbf{x})$ are the *experts*, and the mixing coefficients $\pi_k(\mathbf{x})$ are known as *gating* functions [13].

ME Model Adaptation. Importantly, this initial research involves an adaptation of the ME model, where expert predictions are scaled to be on $[-1, +1]$ (see Eq. 2), and hence cannot be considered true probabilities. For this reason, $p(\mathbf{t}|\mathbf{x})$ also falls on $[-1, +1]$. In this scaling, values approaching -1 indicate groundplane predictions of increasing confidence, values approaching $+1$ indicate obstacle predictions of increasing confidence, and values approaching 0 indicate increasing uncertainty.

This modification is motivated by numerical considerations, as shown by the following scenario. Consider some test point $\mathbf{x}$ for which an expert predicts fully uncertain output, e.g. when $\mathbf{x}$ lies on that expert's decision boundary. Numerically, the uncertain expert prediction must be propagated, regardless of applicability (mixing coefficient). Scaling the expert output on $[-1, +1]$ accommodates this requirement, where uncertain expert output is represented as 0. Otherwise, if expert output were on $[0, 1]$ where uncertain output is represented as 0.5, the final prediction will be distorted by the applicability estimate falling on $[0, 1]$.

In the future, more sophisticated evolutions of our approach will eliminate the need for this adaptation in part by the inclusion of a fully uncertain, generic component that models everything the other components fail to model.

2.2 Experts: Logistic Regression Models

The experts in our technique are *logistic regression* models [14], a linear classification method common in statistics and machine learning, appropriate for predicting the certainty of a binary outcome. Moreover, they are very efficient to compute on large-scale data, motivating their use in the real-time scenarios considered here.

Given a data instance $\mathbf{x}$ and associated model weights $(\mathbf{w}, b)$, logistic regression calculates a continuous probability of the positive output class y for some test instance $\mathbf{x}$ according to the following probability model:

$$P(y = \pm 1 | \mathbf{x}, \mathbf{w}, b) = \frac{1}{1 + \exp\left[-y(\mathbf{w}^T \mathbf{x} + b)\right]} \ , \tag{2}$$

where $\mathbf{w}$ and b are estimated when training the model by minimizing the negative log-likelihood on training data [15].

2.3 Mixing Coefficients: Gaussian Density Models

The mixing coefficients in our technique are determined by Gaussian density models fit to training data when training the expert. When training expert k, a single multivariate Gaussian model $\mathcal{G}_{k,c}$ is learned for each class c of training data from the current image, using the sample mean and covariance of that data. During terrain segmentation, for each test point (i.e., feature vector corresponding to a pixel in the image) $\mathbf{x}$, the mixing coefficients for model k are determined by the response of $\mathcal{G}_{k,c}$ at $\mathbf{x}$:

$$\mathcal{G}_{k,c}(\mathbf{x}|\boldsymbol{\theta}_{k,c}) = \frac{1}{(2\pi)^{d/2} \left|\boldsymbol{\Sigma}'_{k,c}\right|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{k,c})^T (\boldsymbol{\Sigma}'_{k,c})^{-1} (\mathbf{x} - \boldsymbol{\mu}_{k,c})\right] , \tag{3}$$

where $\mathbf{x}$ is a d-dimensional feature vector, $\boldsymbol{\mu}_{k,c}$ is a d-dimensional mean vector, $\boldsymbol{\Sigma}'$ is a scaled $d \times d$ covariance matrix, and $|\boldsymbol{\Sigma}'|$ denotes the determinant of $\boldsymbol{\Sigma}'$. $\boldsymbol{\theta}_{k,c}$ is a parameter set comprising:

$$\boldsymbol{\theta}_{k,c} = \{\boldsymbol{\mu}_{k,c}, \boldsymbol{\Sigma}_{k,c}, \alpha\} \ , \tag{4}$$

where $\boldsymbol{\mu}_{k,c}$ and $\boldsymbol{\Sigma}_{k,c}$ are the sample mean and covariance, respectively, of the training data used to fit $\mathcal{G}_{k,c}$. Finally, $\boldsymbol{\Sigma}_{k,c}$ is subject to scaling by some factor α and application of additive white Gaussian noise ε with mean μ_n and variance σ_n^2:

$$\boldsymbol{\Sigma}'_{k,c} = \alpha \left[\boldsymbol{\Sigma}_{k,c} + \varepsilon \right], \quad \text{where} \;\; \varepsilon \sim \mathcal{N}(\mu_n, \sigma_n^2) \; . \tag{5}$$

Gaussian Noise $\boldsymbol{\varepsilon}$. In implementation, we found that the covariance $\boldsymbol{\Sigma}$ was often singular, and hence not invertible, for feature data typical in this domain; this has been encountered before in similar scenarios when $d \ll N$ [16]. While numerically stable, computing the pseudoinverse was found to yield unsatisfactory results in the response of Eq. 3 for our input. Our solution, known in the literature [16], was to apply a small amount of additive white Gaussian noise to the feature data such that the inverse of the resulting covariance matrix would be defined (Eq. 5). We define this noise ε to be $\sim \mathcal{N}(\mu_n, \sigma_n^2)$, set $\mu_n = 0$, and take an adaptive approach for σ_n^2: starting at 0.01, it is increased by 0.01 until the resulting scaled covariance matrix becomes invertible.

Covariance Scaling Parameter $\boldsymbol{\alpha}$. In initial experimentation with the ME technique, we observed correct behavior in terms of strong response of Eq. 3 near the Gaussian peak (sample mean $\boldsymbol{\mu}_k$), and monotonically decreasing values as $|\mathbf{x} - \boldsymbol{\mu}_k|$ increased. However, the cutoff was too sharp, with most of the output distributed below a value of 0.1. The intuition is that the peak of the density model was too steep (i.e., the distribution was too peaked). Our solution was to scale the covariance $\boldsymbol{\Sigma}$ by some factor α in order to have a more gradual decline in the response of Eq. 3 for increasing $|\mathbf{x} - \boldsymbol{\mu}_k|$. We used $\alpha = 8.0$ in the experiments, which was determined ad-hoc from a sensitivity analysis. In a more sophisticated approach, it is natural that α be data-driven; we will investigate this possibility in future work.

Scaling of the Density Output. The response of the density output of $\mathcal{G}_{k,c}$ in Eq. 3 is scaled such that the value at the peak of the Gaussian distribution (i.e., at the mean $\boldsymbol{\mu}_{k,c}$) is 1. Hence, the mixing coefficient $\pi_{k,c}(\mathbf{x})$ will be maximal when $\mathbf{x}$ is close to the sample mean of the data used to train the model. This scaling is achieved by dividing the response of Eq. 3 by its output at the sample mean of the data used to train the model:

$$\pi_{k,c}(\mathbf{x}|\boldsymbol{\mu}_k) = \frac{\mathcal{G}_{k,c}(\mathbf{x})}{\mathcal{G}_{k,c}(\boldsymbol{\mu}_{k,c})}. \tag{6}$$

Determination of $\boldsymbol{\pi_k}(\mathbf{x})$ from $\boldsymbol{\pi_{k,c}}(\mathbf{x})$. The final mixing coefficient $\pi_k(\mathbf{x})$ is simply the *maximum* of the C single-class density model outputs $\pi_{k,c}$ at $\mathbf{x}$:

$$\pi'_k(\mathbf{x}) = \max_c \left[\pi_{k,c}(\mathbf{x}) \right]. \tag{7}$$

Alternatives for future investigation include determining $\pi_k(\mathbf{x})$ as either the simple or the weighted mean of the C density model outputs $\pi_{k,c}(\mathbf{x})$, instead of the maximum.

2.4 Modifications Permitting Ensemble Abstinence

With a traditional mixture model, although individual models can abstain from making predictions, the entire ensemble cannot, because $\sum_k \pi_k(\mathbf{x}) = 1$. This effectively assumes that the correct model, or combination of models, exists in the ensemble for a given data instance $\mathbf{x}$, which we do not wish to assume. In particular, we wish to allow the entire *a posteriori* ensemble output to abstain for certain $\mathbf{x}$.

Consider the case where all models in the ensemble predict $+1$ for $\mathbf{x}$, but whose mixing coefficients at $\mathbf{x}$ are all 0.1. In the traditional ME approach, the mixing coefficients would be scaled to sum to 1, resulting in a full confidence $+1$ output. This output is not reflective of the underlying models' low applicability at $\mathbf{x}$.

On the other hand, if no scaling is done, undesirable behavior will result. Consider if there were 10 experts in the ensemble, all predicting $+1$ at $\mathbf{x}$, each with mixing coefficients of 0.1. With no scaling of the mixing coefficients, their cumulative ME sum would result in a final, full-confidence prediction of $+1$. In the same scenario but with 20 such experts, without scaling the final ME output would be greater than 1; some scaling is needed to bound the output.

In short, a scaling approach is needed that bounds final ME ensemble output, but also allows the ensemble to abstain if its underlying experts all wish to abstain. Our solution is to scale the sum of the K mixing coefficients at $\mathbf{x}$ to the K experts' *mean density model response* at $\mathbf{x}$, such that $\sum_k \pi_k(\mathbf{x}) = \frac{1}{K}\sum_k \pi'_k(\mathbf{x})$. This is achieved by dividing each unscaled mixing coefficient $\pi'_k(\mathbf{x})$ by the total number of experts K:

$$\pi_k(\mathbf{x}) = \frac{\pi'_k(\mathbf{x})}{K} \ . \tag{8}$$

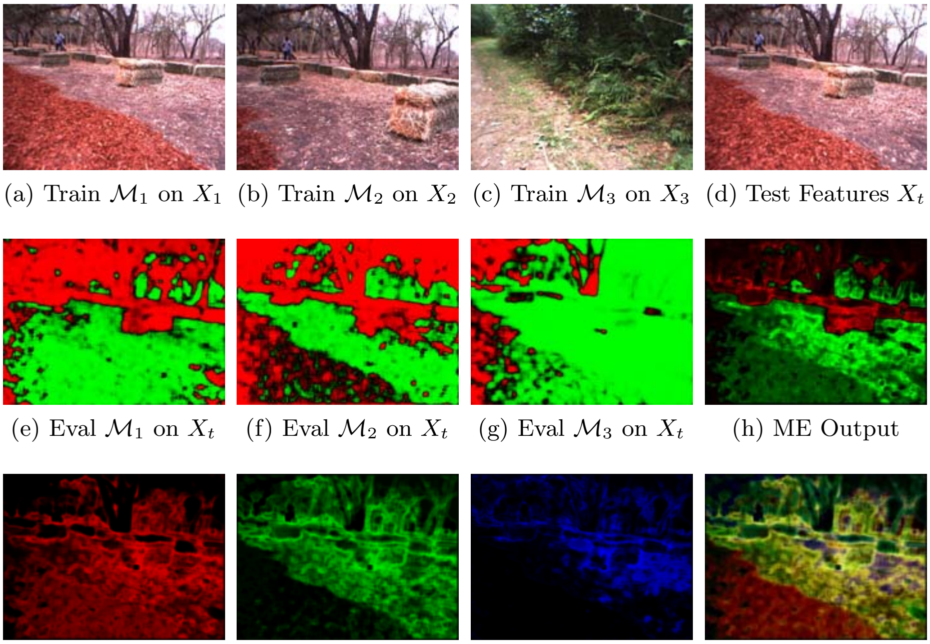
Thus, if the individual experts all have low applicability estimates, the final ensemble output will be reflective of this and will have an appropriate low-confidence response.

2.5 Conceptual Overview of the ME Approach Applied to the Domain

A conceptual overview of the approach as applied to terrain segmentation is shown in Fig 3. This is also the basis of the experimental approach discussed in Sec. 3.1.

Consider an ensemble composed of three models $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$. These models are trained on the corresponding feature data sets X_1, X_2, and X_3, extracted from images I_1, I_2, and I_3, respectively (shown in Figs. 3a–3c). A fourth image I_t, and its associated feature data set X_t, is the current target image requiring terrain classification (Fig. 3d). (Note that I_1 and I_t are similar, but not identical, frames.)

Each of the three linear models $\{\mathbf{w}_k, b_k\}$ in the ensemble is applied to X_t (Eq. 2); their corresponding terrain predictions are shown in Figs. 3e–3g. In the ME approach, X_t is also evaluated through the two density models $\{\mathcal{G}_{k,c=1}, \mathcal{G}_{k,c=2}\}$



(a) Train $\mathcal{M}_1$ on X_1 (b) Train $\mathcal{M}_2$ on X_2 (c) Train $\mathcal{M}_3$ on X_3 (d) Test Features X_t

(e) Eval $\mathcal{M}_1$ on X_t (f) Eval $\mathcal{M}_2$ on X_t (g) Eval $\mathcal{M}_3$ on X_t (h) ME Output

(i) $\pi_1(X_t)$ (j) $\pi_2(X_t)$ (k) $\pi_3(X_t)$ (l) Composite $\boldsymbol{\pi}(X_t)$

Fig. 3. Conceptual overview of ME approach for terrain segmentation

learned from each the three training images, yielding the mixing coefficients π_k (Eqs. 3–8), shown in Figs. 3i–3k. The *composite coverage* $\boldsymbol{\pi}$ of the three experts is shown in Fig. 3l. The final terrain classification output from the ME approach (Eq. 1) is given in Fig. 3h.

3 Experimental Evaluation

3.1 Approach

Baseline Algorithm. We compare the ME approach to a basic *unweighted average* baseline method [3], in which the terrain classification from each model in the ensemble is averaged together to arrive at the final terrain classification.[3]

Data Sets. The evaluation is performed using six hand-labeled natural data sets taken from the domain, recently contributed by the authors [5] and made publicly available [17]. Each dataset consists of a 100-frame hand-labeled image sequence. The images were manually labeled, with each pixel being placed into one of three classes: Obstacle, Groundplane, or Unknown; further details are available [3].

[3] This can be seen as a general case of *majority voting*, appropriate when individual expert output is on $[0, 1]$ and final ensemble output on $[0, 1]$ is also desirable.

Method. The experimental method follows the conceptual approach outlined previously in Sec. 2.5 and Fig. 3. We conducted a series of 3,000 randomized experiments, drawing one training image at random from each of the six datasets for each experiment. A testing image, with known ground truth labeling, was also drawn. Six models were learned on the training images, and each model was then applied to the test image. To obtain the output of the unweighted average baseline method, the terrain classification was combined using a simple average. For the ME approach, mixing coefficients were determined, and then combined with the linear model output according to the ME model (Eq. 1) to arrive at the final ME terrain segmentation.

Evaluation. We evaluated our approach only on the pixels in that portion of the image occurring in the far field, i.e., greater than 10 m but less than 100 m from the robot. We used standard binary classification accuracy (ACC), and since the data is roughly 3:1 skew, we also report ACC for baselines of predicting all of the same class. We report the mean ACC and std. dev. across all randomized experiments.

3.2 Results and Discussion

Experimental data is provided in Table 1; sample output is shown in Fig. 4.

Statistical Analysis. The scores shown in Table 1 represent mean values of scores from 3,000 randomized experiments. Within each experiment, the scores for each method were determined by evaluation on the same test image.

Thus, a dependent-samples analysis can be performed to determine whether or not the difference in performance among algorithms is statistically significant. Because the distribution of the differences between the paired samples was found to be non-normal, the equivalent nonparametric test statistic, the *Sign* test [18], was used.

This test provided sufficient statistical evidence to infer that the medians of the differences between each population are not 0, significant at the 95% confidence level.[4] We conclude that the ME approach outperforms the baseline approach in this evaluation.

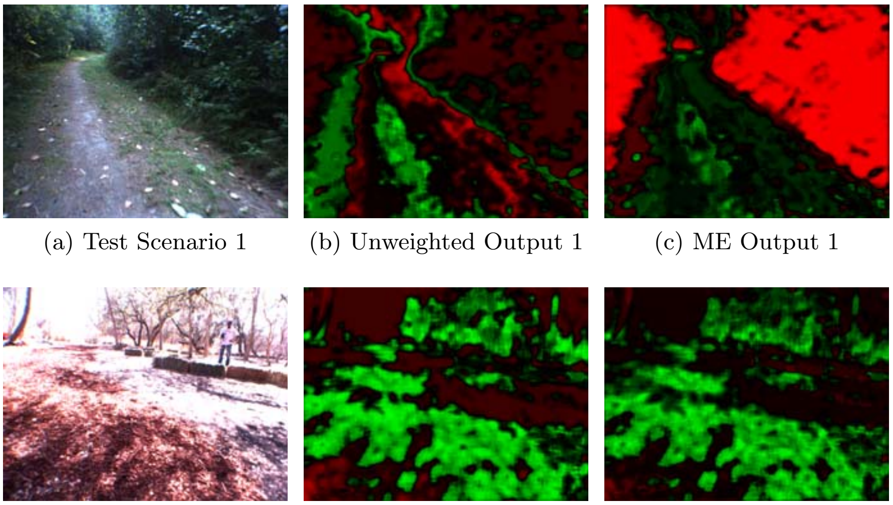
Discussion. These results illustrate the benefits of allowing individual models to abstain. In one scenario (Figs. 4a–4c), the ME method assigns low weight to the incorrect models, allowing the correct model (i.e., the one model of the six that matches the scenario of the target image I_t) to carry the most weight in the final ensemble output. In a different scenario (Figs. 4d–4f), for certain parts of the image, none of the models in the ensemble were applicable, allowing "uncertain" regions to pass through to the final output. In both cases, if models were blindly applied without regard to local accuracy, incorrect terrain predictions would have been made.

[4] Although the standard deviations for the scores appear high, they are similarly high for each group of samples, and are due to within-sample variance.

Table 1. ME Algorithm Performance vs. Baseline

Algorithm	**Score (ACC)**[a,b]
Mixture of Experts (Local)	0.759 ± .21
Unweighted Average (Non-Local)	0.683 ± .24
Predict all OBSTACLE	0.726 ± .16
Predict all GROUNDPLANE	0.274 ± .16

[a] Binary accuracy, thresholded at 0.5.
[b] Mean and standard deviation of 3,000 randomized experiments.


(a) Test Scenario 1 (b) Unweighted Output 1 (c) ME Output 1

(d) Test Scenario 2 (e) Unweighted Output 2 (f) ME Output 2

Fig. 4. Experimental snapshots from two scenarios

In our experiments, we observed the technique to be real-time in our parallel implementation, when run on hardware with computational performance comparable to that of a typical robotic platform. An in-depth study of the computational characteristics of the proposed ME approach is an area for future work.

4 Conclusions and Future Work

In this paper, we presented a novel adaptation of the mixture of experts model for determining model applicability, and applied this technique to the terrain segmentation problem in the outdoor autonomous robot navigation domain. This method accommodates the key domain constraints associated with near-to-far learning for autonomous robot navigation: models are learned over time; model training and evaluation must be performed in real time; training data is not kept once a model has been trained, due to storage limitations; and training data, derived from stereo, may not be available for the target image requiring terrain segmentation.

We evaluated our approach, a local method, against a non-local, unweighted average as a baseline in a statistically significant evaluation, and concluded that the proposed mixture of experts approach outperforms the unweighted average baseline in far-field terrain prediction performance. In particular, the ME approach's inherent ability to permit individual experts to abstain from making strong predictions on a local (i.e., pointwise) basis allows for the more applicable models' predictions to carry the most weight in the final ensemble prediction.

Future work. We identify three key areas of ongoing future work. First, we plan to conduct a more in-depth experimental evaluation, varying key factors such as data-driven methods for determining α and variants of the *class density combination* function (Eq. 7) to determine impact on performance. This analysis will also involve comparison against another local method, described earlier in the paper, which takes an alternative approach for estimating local accuracy [12]; the computational performance of each will be considered.

Second, we will investigate posing this problem more formally in the hierarchical mixtures of experts (HME) context [7], using EM to fit model parameters, and making use of validation data from the target image to influence mixing coefficients.

Finally, while the notion of a mixture model with the ability to abstain appears useful, the theory behind its implementation needs to be improved. The inclusion of an uncertain, generic component should greatly improve the theoretical expression.

Acknowledgments. The authors gratefully acknowledge the contribution of Sandia National Laboratories, the National Science Foundation, the DARPA LAGR program, and the anonymous referees' insightful comments. Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

References

1. Jackel, L., Krotkov, E., Perschbacher, M., Pippine, J., Sullivan, C.: The DARPA LAGR program: Goals, challenges, methodology, and Phase I results. Journal of Field Robotics 23(11-12), 945–973 (2006)
2. Howard, A., Turmon, M., Matthies, L., Tang, B., Angelova, A., Mjolsness, E.: Towards learned traversability for robot navigation: From underfoot to the far field. Journal of Field Robotics 23(11–12), 1005–1017 (2006)
3. Procopio, M.J., Mulligan, J., Grudic, G.: Learning terrain segmentation with classifier ensembles for autonomous robot navigation in unstructured environments. Journal of Field Robotics 26(2), 145–175 (2009)
4. Procopio, M.J., Mulligan, J., Grudic, G.: Long-term learning using multiple models for outdoor autonomous robot navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 2007, pp. 3158–3165 (2007)

5. Procopio, M.J., Mulligan, J., Grudic, G.: Learning in dynamic environments with Ensemble Selection for autonomous outdoor robot navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), September 2008, pp. 620–627 (2008)
6. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Computation 3(1), 79–87 (1991)
7. Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and the EM algorithm. Neural Computation 6, 181–214 (1994)
8. Jacobs, R.A.: Methods for combining experts' probability assessments. Neural Computation 7(5), 867–888 (1995)
9. Woods, K., Kegelmeyer, W., Bowyer, K.: Combination of multiple classifiers using local accuracy estimates. IEEE Trans. Pattern Analysis and Machine Intelligence 19(4), 405–410 (1997)
10. Sato, M.A., Ishii, S.: On-line EM algorithm for the normalized gaussian network. Neural Computation 12(2), 407–432 (2000)
11. Moody, J., Darken, C.J.: Fast learning in networks of locally-tuned processing units. Neural Computation 1(2), 281–294 (1989)
12. Grudic, G., Mulligan, J., Otte, M., Bates, A.: Online learning of multiple perceptual models for navigation in unknown terrain. In: FSR 2007: Proceedings of the International Conference on Field and Service Robotics (2007)
13. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, New York (2006)
14. Cox, D.R., Snell, E.J.: Analysis of Binary Data, 2nd edn. Chapman Hall, London (1989)
15. Lin, C.J., Weng, R.C., Keerthi, S.S.: Trust region Newton method for large-scale logistic regression. J. Mach. Learn. Res. 9, 627–650 (2008), http://www.csie.ntu.edu.tw/~cjlin/liblinear/
16. Grudic, G., Mulligan, J.: Outdoor path labeling using polynomial mahalanobis distance. In: Proceedings of Robotics: Science and Systems (RSS), Philadelphia, PA (2006)
17. Procopio, M.J.: Hand-labeled DARPA LAGR datasets (2007), http://ml.cs.colorado.edu/~procopio/labeledlagrdata/
18. Hollander, M., Wolfe, D.A.: Nonparametric statistical methods, 2nd edn. Wiley-Interscience, New York (1999)

Consistency Measure of Multiple Classifiers for Land Cover Classification by Remote Sensing Image

Peijun Du, Guangli Li, Wei Zhang, Xiaomei Wang, and Hao Sun

Department of Remote Sensing and Geographical Information Science,
China University of Mining and Technology,
Xuzhou City Jiangsu Province 221116, P.R. China
dupjrs@cumt.edu.cn, dupjrs@gmail.com

Abstract. Nowadays, multiple classifier system is widely used for land cover classification by remote sensing imagery. The performance of combined classifier is closely related to the selection of member classifiers, so it is necessary to analyze the diversity and consistency of member classifiers. In our study, consistency measures are studied and experimented from three levels: general consistency measure, binary prior measure and consistency of errors, and the result shows that it is feasible to find the effective set of member classifiers by some consistency measures. In land cover classification by remotely sensed classifiers, we can select optimal member classifiers by integrating different consistency criterions.

Keywords: consistency measure, multiple classifier combination, land cover classification, remote sensing, weighted count of errors and correct results (WCEC).

1 Introduction

Land cover is a fundamental variable influencing and linking many factors of the human and physical environment [1]. Remote sensing has provided an efficient way of data acquisition for land use administration and land cover mapping, in which land cover classification by remote sensing image is the basic work. Recently, multiple classifier system (MCS) has been applied to land cover classification, because the performances of various classifiers on a specific land cover classification are often dissimilar and it is necessary to combine those classifiers. Benediktsson applied some MCS methods (boosting, bagging, consensus theory and random forests) to multi-source remotely sensed data for land cover classification [2]. A good multiple classifier system depends on not only combination rules, but also member classifiers selected from a classifier pool [3]. Nowadays, how to select optimal set of member classifiers has been raised in attention as this is one of the critical issues to the success of MCS. Kang and Lee reported some strategies for selecting multiple classifiers [4]. Kang proposed some measures for selecting classifiers, including closeness (MC), conditional entropy (CE) and minimization of mutual information (mMI) [3].

In this paper, we are less concerned about the scheme of combination, but mainly focus on the methods of selecting optimal member classifiers from a classifier pool. According to Michalis, consistency can inhibit the gains obtained regardless of the method used to combine classifiers [5]. Consistency is an index to measure the degree of similarity for different investigators (methods) [6]. Generally speaking, high consistency of classifiers is not beneficial for improving overall accuracy because classifiers with less consistency could provide complementary each other. Consistency measure in this paper is based on various land cover classification results by different classifiers for one specific remotely sensed image.

In the study, consistency is respectively measured from three levels: general consistency measure, binary prior measures and consistency of errors. Experiments are carried out to show the relationship between consistency and the performance of combined classifiers, and to compare the criterions of consistency measures for selecting optimal member classifiers. It should be pointed that the classifiers used for land cover classification are dependent mutually, as they use the same training samples and feature sets.

2 Consistency Measures

Aksela defined diversity measures from different viewpoints, which can be grouped into three categories based on various sets of values from which the measures are calculated [7]. Consistency measures from the three aspects are as follows: (1) general consistency measure by kappa statistics which uses information on all class labels; (2) kappa statistic and double-fault used as binary prior measures which employ prior knowledge on the correctness of the classification results of classifiers; (3) consistency of errors by kappa statistic and weighted count of errors and correct results (WCEC) that use all class labels as well as prior knowledge on correctness. Consistency measure in this paper is just based on test samples, not on the whole classification results.

Assuming that m ($m \geq 2$) classifiers are used for land cover classification by one remotely sensed image, consistency is measured based on the classification results for n test samples. There are k($k \geq 2$) possible outputs of each classifier, while the number of possible class labels is b, different from k, y_{ij} represents the number of classifiers whose outputs are j for the ith class test samples, $i = 1,\dots,n$; $j = 1,\dots,k$.

2.1 General Consistency Measure

General consistency measure does not use test samples to evaluate the correctness of the classification results produced by multiple classifiers, but only estimate the consistency from the outputs of every classifier based on class label [7]. In this case, the number of possible outputs of classifiers, k, is just the same with that of possible class labels b.

Kappa (κ) statistic is firstly put forward by Cohen in 1960, which is a measure of consistency between two surveyors after chance being corrected [6, 8]. Kappa statistic has been used for assessing the accuracy of classification of remote sensing data based on the error matrix [9]. This in effect means checking the degree of consistency between the classification results of the classifier and the reference data [5].

Cohen's kappa is suited for two classifiers. If there are more classifiers, the following formula is adopted.

The equation of kappa statistic is [10, 11]:

$$\kappa = 1 - \frac{nm^2 - \sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}^{\ 2}}{nm(m-1)\sum_{j=1}^{k} p_j q_j} \tag{1}$$

Where, $i = 1,\ldots,n$; $j = 1,\ldots,k$, $p_j = \sum_{i=1}^{n} y_{ij} \Big/ nm$ represents the overall proportion of outputs of classifiers in category j, and $q_j = 1 - p_j$. For every test sample, $\sum_{j=1}^{k} y_{ij} = m$.

A value of kappa below 0.40 is considered to represent poor agreement beyond chance, values between 0.40 and 0.75 indicate fair agreement, and values beyond 0.75 indicate excellent agreement [10].

In principle, the kappa value is minimized to select the optimal subset of classifiers to improve the diversity of member classifiers.

2.2 Binary Prior Measure

This method uses the "0/1" outputs of classifiers based on test samples [11]. That is to say, when a test sample i is correctly classified by the lth classifier, its output to the test sample is defined as 1, otherwise 0. So classifiers have two possible outputs to a specific ground truth, correctness or incorrectness. In the case, k is equal to 2, not the same as b.

Furthermore, as n is the total number of test samples, for a two-classifier measure, n^{11} is used to denote the number of samples correctly classified by both classifiers, n^{00} is the number of samples incorrectly classified by both classifiers, and n^{10} and n^{01} the number of samples when just correctly classified by the first or second classifier respectively. Naturally n=$n^{11} + n^{00} + n^{10} + n^{01}$.

The following are two ways of binary prior measures, double fault and kappa.

2.2.1 Double Fault (DF)

DF is defined as the proportion of the cases that has been misclassified by both classifiers [12]. It is a pairwise measure, and for the whole combination the averaged value over all pairs of classifiers is computed. In theory, the mean pairwise value is minimized for selecting the subset of classifiers.

The equation of Double fault (DF) is[7]

$$\text{DF} = \frac{n_{00}}{n} \tag{2}$$

2.2.2 Kappa (κ)

The definition of the kappa is the same with that in 2.1, with the difference that the possible outputs of the classifiers are either 0 or 1. In the case the value of k is two. The κ value calculated by Equation (1) reflects the degree of consistency of the classifiers to be used based on prior knowledge, which is a non-pairwise method.

It should be noted that if the classifiers used for combination make errors to one test sample, they all agree on the "0"output, although they actually disagree on the specific class the test sample belong to[5].

In principle, the kappa is minimized for selecting the set of classifiers.

2.3 Consistency of Errors

Although classifiers all make errors to a specific test sample sometimes, the errors may be different according to classification results. Consistency on the same incorrect class is harmful for classifier combination. The method of consistency measure not only takes the prior knowledge into consideration, but also need to use the class labels.

2.3.1 Kappa (κ)

This κ is based on the consistency of errors, also calculated by Equation (1). For two classifiers, we respectively collect their own class labels for test samples when incorrectly classified by both of them. κ can be computed with the two sets of class labels.

In theory, mean pairwise κ value is minimized for selecting the set of classifiers.

2.3.2 Weighted Count of Errors and Correct Result (WCEC)

Weighted count of errors and correct result (WCEC) takes both correct and incorrect results into consideration and gives suitable weight on them [7]. If there are two classifiers, the Equation is

$$WCEC = n^{11} + \tfrac{1}{2}\left(n^{01} + n^{10}\right) - n^{00}_{different} - 5n^{00}_{same} \tag{3}$$

Where, $n^{00}_{different}$ stands for the number of samples incorrectly classified by both classifiers with different errors, and n^{00}_{same} represents the number of samples incorrectly classified by both classifiers, but with the same classification results.

For more than two classifiers, the mean of the pairwise counts is calculated. In theory, the WCEC is maximizing for selecting the optimal set of classifiers.

General consistency measure, binary prior measure, and consistency of errors all use κ values, so we use κ_1 , κ_2 and κ_3 to distinguish the different results by the three types of measures, that is to say, κ_1 represents κ value calculated by general consistency measure, K_2 represents that of binary prior measure and K_3 represents that of consistency of errors.

3 Classifier Combination

There are many methods for combing classifiers in pattern recognition. Based on the output types of individual classifier, the multiple classifier combination methods can be classified into three categories: combination at abstract level, rank level and measurement level [18]. In this paper, abstract level fusion is used, and the most popular algorithm is voting rule. The voting rule is based on an assumption that population's decision is superior to the individual's decision [15]. There are different voting rules [16]. In the majority voting rule, label that received more than half of the votes is taken as the finial classification output. In the conservative voting rule, only if all the member classifiers are consensus with the same class label, the sample can be designated as the label. Weighted vote rule is also widely used, for the performances of classifiers differ to different land cover types [19]. In our experiment, the weights are designed according to different user's accuracy of land cover classes, as overall accuracy is too general to reflect the classification performances of different land cover classes and producer's accuracy is used for producers, less suitable for predicting the final class labels. Dempster-Shafer (DS) evidence theory rule of combination is also used to implement the classifier combination [17].

4 Experiment and Analysis

4.1 Land Cover Classification

4.1.1 Study Area and Data Used

An experiment was performed using Landsat ETM+ multispectral image, captured on 3 April, 2001, with the spatial resolution of 28.5m. The spatial scope of the study area is: latitude 34°13′N to 34°19′ N and longitude 117°07′E to 117°13′ E. The data used for classification consists of six bands (the thermal band and panchromatic band were excluded) and has a size of 400*400 pixels.

4.1.2 Remote Sensing Classifiers

Decision tree classification (DTC) is an effective method of classifying data set, and can provide good decision support capabilities. For many problems of classification where large datasets are used and the information contained is complex, even may contain errors, decision trees provide a useful solution [21].

Support Vector Machine (SVM) classification is based on statistical learning theory and it belongs to the class of non-parametric classification methods [22]. The criterion of SVM is Structural Risk Minimization (SRM) and it can learn "good" classification hyperplane in high dimension feature space [23].

Back Propagation Neural Network (BPNN) algorithm is underpinned by a gradient descent algorithm that is used to modify the network weights to maximise performance, using some criterion function [24].

Minimum Distance Classifier (MD) calculates the distance between each pixel of remotely sensed image and the mean vector of every class, and then the pixel will be classified to the class with the minimal distance.

Spectral Angle Mapper (SAM) is a physically-based spectral classification that uses an n-dimensional angle to match pixels to reference spectra [25]. The algorithm determines the spectral similarity between test spectra and reference spectra.

4.1.3 Classification Implementation and Result Comparison

Considering regional natural condition and ground characteristic, land cover is classified into five types: water, forest, built-up land, agricultural land and grass land. Training and test samples are also selected for accuracy evaluation and further consistency measure. There is almost no overlap between the training samples and test samples in order to decrease the correlation of them. Land cover classification by SVM, DTC, BPNN, MD and SAM are implemented on ENVI 4.3, using the same training samples. The SVM parameters are determined by gridding search model. In our experiment, radial basis function is adopted. Penalty parameter C is 150 and γ in kernel function is 0.170. The feature sets used for classification are identical for different classifiers. Land cover classification results by single classifier are shown in Fig.1. As shown in Table 1, it is clear that for this data set, MD is the worst classifier, giving a total classification accuracy of just 80.40% for test data. SVM is seen to be the best classifier with an overall accuracy of 92.42% for the same test set. The performances of the classifiers are different according to accuracy comparison, so it is necessary to combine classifiers to improve classification accuracy.

Table 1. Comparison of classification accuracies of MLC, SVM, BPNN, SAM and MD

Classifier	SVM	DTC	BPNN	SAM	MD
Overall accuracy/%	92.42	89.58	87.34	84.37	80.40
kappa	0.91	0.87	0.84	0.80	0.76

4.2 Consistency Measure and Analysis

Each classifier is given a serial number in terms of overall accuracy, so the serial numbers of SVM, DTC, BPNN, SAM and MD are respectively 1, 2, 3, 4 and 5. The number of selected member classifiers is three, so there are 10 possible combinations. As shown in Table 2, the selection criterions are computed for every combination. Besides, three multiple classifier combination schemes including weighted vote, DS evidence theory and majority voting are carried out, and the overall accuracies are also shown in Table 2.

The degree of consistency can be analyzed from Table 2. The average values of K_1, K_2 and K_3 for all possible combination sets are respectively 0.824, 0.460 and 0.555, calculated based on different levels of consistency. The average value of K_1

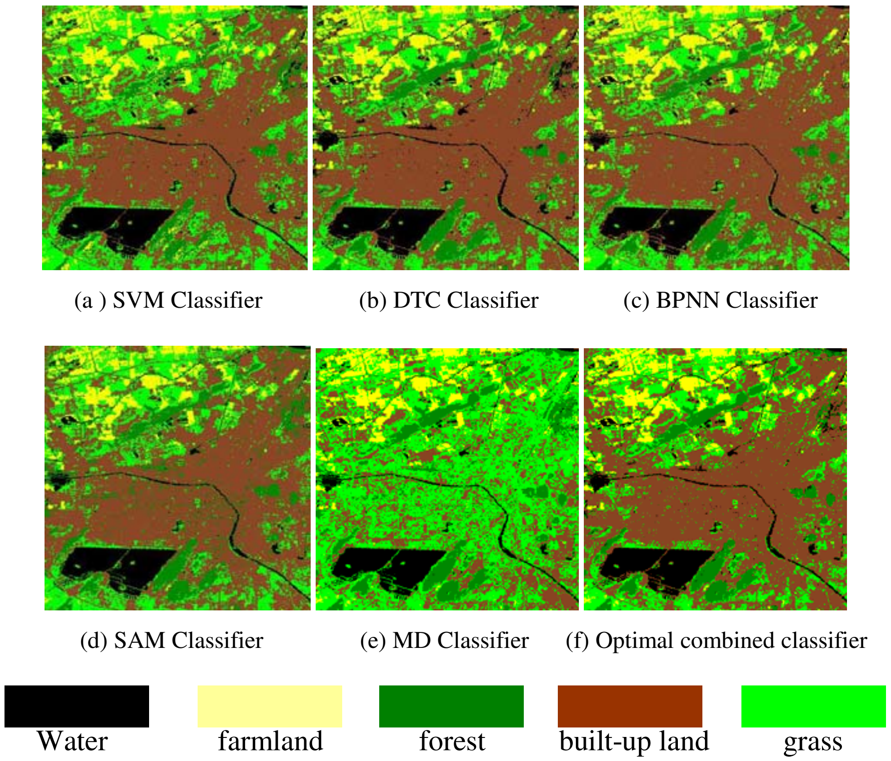


Fig. 1. Land cover classification results of ETM+ by single classifier and combined classifier

Table 2. Comparison of the performances of different criterions of consistency measure for selecting optimal member classifiers

Members	κ_1	DF	κ_2	WCEC	κ_3	Weighted vote(%)	DS (%)	majority (%)
1,2,3	0.868	0.057	0.515	0.680	0.490	91.191	91.211	91.315
1,2,4	0.856	0.075	0.515	0.647	0.563	89.330	89.330	89.578
1,2,5	0.796	0.086	0.326	0.681	0.407	92.556	92.622	92.928
1,3,4	0.892	0.084	0.628	0.514	0.818	89.826	89.825	89.950
1,3,5	0.839	0.098	0.463	0.532	0.551	91.687	91.687	91.687
1,4,5	0.813	0.110	0.423	0.529	0.751	91.315	91.310	91.563
2,3,4	0.842	0.079	0.558	0.577	0.384	87.717	87.717	87.841
2,3,5	0.770	0.060	0.377	0.632	0.405	89.578	89.578	90.074
2,4,5	0.764	0.069	0.334	0.586	0.492	88.089	88.120	88.709
3,4,5	0.804	0.086	0.458	0.441	0.692	87.717	87.717	88.089

shows that land cover classification by multiple classifiers is excellent agreement in the general level. The average value of κ_2 indicates when correctness of classification is taken into account, the classification results by multiple classifiers are fairly agreement. It can be seen that the degree of consistency of errors evaluated by κ_3 is not low, which is harmful for classifiers combination. The degrees of consistency vary with the levels on which consistency is defined.

Consistencies of different sets of classifiers for combination are respectively evaluated by κ_1, κ_2, κ_3, DF, and WCEC. The different values of the criterions are related to different classification accuracy indicators of combined classifiers. We respectively choose the optimal sets of combination by minimizing κ_1, κ_2, κ_3, DF and maximizing WCEC and obviously the selected sets are different. The general consistency measure using κ_1 could not find very effective member classifiers sets because the criterion does not take into account the difference between errors and correct decisions. The optimal set of member classifiers selected by DF is {1, 2, 3}, with the overall accuracy of 91.191% by weighted vote rule, which is obviously better than that of the combined classifiers selected by κ_1. The set of member classifiers selected by κ_3 is {2, 3, 4}, with poor overall accuracy of 87.717% by weighted vote combination. WCEC and κ_3 both find the best selection of {1, 2, 5}, with a total classification accuracy of 92.556% by combing classifiers using weighted vote rule, which is just the one with the highest overall accuracy in all the possible combinations. The sixth image in Fig.1 is the classification result by combined classifier of the set of {1, 2, 5}. It also can be seen from Table 2 when combination methods change the criterions also take effect for selecting members for combining classifiers.

It can be seen from Table 2 that if the selection of member classifiers is good enough, such as {1, 2, 5}, it may give an overall accuracy higher than that of the excellent singe classifier. The following analysis is based on the values of κ_2. It seems that classifiers with high classification accuracy agree much mutually, but their combination may not improve overall accuracy much. On the opposite, the classifiers with small degree of consistency may have great potential to improve classification accuracy. By combining classifiers, sometimes it may not improve the performance of the best classifier, but beneficial for poor classifiers. Although the MD classifier is the poorest classifier in terms of accuracy, the optimal combination contains it. So when we perform classifier combination, the inaccurate classifier should not be excluded.

5 Conclusions

In this paper, consistency measure is used for selecting optimal remotely sensed member classifiers to be combined for land cover classification. It can be seen that some consistency criterions are able to choose effective sets of member classifiers. Sometimes they don't select the truly optimal set of member classifiers in terms of overall accuracy, as in fact, the performance of combined classifier is also affected by the accuracy of individual classifier. The optimal member classifiers selected by different criteria are not the same. Though general consistency measure is widely used for many fields, the results indicate it is not quite suitable for selecting optimal member classifiers, as it does not consider the correctness of land cover classification. Binary prior measure performs better than general consistency measure, for that it takes the correctness of classification into consideration. However, it ignores the different errors made by classifiers and though the outputs of some classifiers for specific samples are all “0”, the true classification results may be different. The third level of consistency measure takes the consistency of errors into consideration. WCEC punishes the consistency of errors and rewards the consistency of correct outputs, and also weights are given for the different kinds of outputs. In addition, other experiments using different pools of classifiers are also carried out. It can be concluded that κ_2, DF and WCEC are beneficial for finding an effective member classifier set. κ_3 sometimes failed to select a good combination, as it only focused on the consistency of errors, ignoring the correct outputs. In land cover classification by remotely sensed data, we can select optimal member classifiers by integrating different consistency criterions.

Acknowledgments. The authors thank the support to this research from National High-Tech 863 Program of China (2007AA12Z162), the Natural Science Foundation of China (40871195), the Program for New Century Excellent Talents in University (NCET-06-0476) and the opening fund of Key Laboratory of Advanced Engineering Surveying of SBSM (ES-SBSM-(07)-01).

References

1. Foody, G.M.: Status of Land Cover Classification Accuracy Assessment. Remote Sensing of Environment (80), 185–201 (2002)
2. Benediktsson, J.A., Chanussot, J., Fauvel, M.: Multiple classifiers system in remote sensing: from basics to recent developments. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 501–512. Springer, Heidelberg (2007)
3. Kang, H.J., Doermann, D.: Selection of Classifiers for the Construction of Multiple Classifier Systems. In: Proceedings of the Eight International Conference on Document Analysis and Recognition, pp. 1194–1198 (2005)
4. Kang, H.J., Lee, S.W.: Experimental Results on the Construction of Multiple Classifiers Recognizing Handwritten Numerals. In: Proc. of the 6th ICDAR, pp. 1026–1030 (2001)
5. Petrakos, M., Benediktsson, J.A.: The Effect of Classifier Agreement on the Accuracy of the Combined Classifier in Decision Level Fusion. IEEE Transactions on Geoscience and Remote Sensing 39(11), 2539–2545 (2001)
6. Shan, B.: Studies on Methods of Evaluating Agreement among Multiple Investigators (Methods of Measurement). Dissertation of Academy of Military Medical Sciences of PLR of China (2006)
7. Aksela, M., Laaksonen, J.: Using Diversity of Errors for Selecting Members of A Committee Classifier. Pattern Recognition (39), 608–623 (2006)
8. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement 20, 37–46 (1960)
9. Congalton, R.: A Review of Assessing the Accuracy of Classifications of Remotely Sensed Data. Remote Sensing of Environment (37), 35–46 (1991)
10. Fleiss, J.L.: Statistical Methods for Rates and Proportions. Wiley, New York (1981)

11. Chen, D., Sirlantzis, K., Dong, H., et al.: On the Relation between Dependence and Diversity in Multiple Classifier Systems. In: Proceedings of International Conference on Information Technology: Coding and Computing, vol. (1), pp. 134–139 (2005)
12. Giancinto, G., Roli, F.: Design of Effective Neural Network Ensembles for Image Classification Purposes. Image Vision Computer (19), 697–705 (2001)
13. Bedau, M., Zwick, M., Bahm, A.: Variance and Uncertainty Measures of Population Diversity Dynamics. Adv. Syst. Sci. Appl (Special issue I), 7–12 (1995)
14. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On Combining Classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
15. Lam, L.: Classifier Combinations: Implementations and Theoretical Issues. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 77–86. Springer, Heidelberg (2000)
16. Bo, Y.C., Wang, J.F.: Combining Multiple Classifiers for Thematic Classification of Remotely Sensed Data. Journal of Remote Sensing 9(5), 557–562 (2005)
17. Han, D.Q., Han, C.Z., Yang, Y.: Combination of Heterogeneous Multiple Classifiers Based on Evidence Theory. In: Proceedings of the 2007 International Conference on Wavelet Analysis and Pattern Recognition, pp. 573–578 (2007)
18. Congalton, R.G., Green, K.: Assessing the Accuracy of Remotely Sensed Data. In: Principles and Practices. Lewis Publishers (1999)
19. Suen, C.Y., Lam, L.: Multiple Classifier Combination Methodologies for Different Output Levels. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, p. 52. Springer, Heidelberg (2000)
20. Han, J.F., Yang, Z.H.: Combined Classifiers and Its Application in Hyperspectral Classification. Journal of Zhengzhou Institute of Surveying and Mapping 24(3), 231–234 (2007)
21. Aitkenhead, M.J.: A co-evolving decision tree classification method. Expert Systems with Applications 1, 18–25 (2008)
22. Jens, K., Simone, N., et al.: Automatic land cover analysis for Tenerife by supervised classification using remotely sensed data. Remote Sensing of Environment 86(4), 530–541 (2003)
23. Nello, C., John, S.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, p. 82. Cambridge University Press, New York (2000)
24. Chai, S., Veenendaal, B., et al.: Backpropagation Neural Network for Soil Moisture Retrieval Using Nafe 2005 Data: A Comparison of Different Training Algorithms. In: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Beijing, vol. XXXVII, Part B4 (2008)
25. Kruse, F.A., Boardman, J.W., et al.: The Spectral Image Processing System (SIPS): Interactive visualization and analysis of imaging spectrometer data. Remote Sensing of Environment 44, 145–163 (1993)

Target Identification from High Resolution Remote Sensing Image by Combining Multiple Classifiers

Peijun Du, Hao Sun, and Wei Zhang

Department of Remote Sensing and Geographical Information Science,
China University of Mining and Technology,
Xuzhou City Jiangsu Province 221116, P.R. China
dupjrs@cumt.edu.cn, dupjrs@gmail.com

Abstract. Target identification from high resolution remote sensing image is a common task for many applications. In order to improve the performance of target identification, multiple classifier combination is used to a QuickBird high resolution image, and some key techniques including selection and design of member classifier, classifier combination algorithm and target identification methods are investigated. After constructing a classifier ensemble composed of five members: maximum likelihood classifier (MLC), minimum distance classifier (MDC), Mahalanobis distance classifier (MHA), decision tree classifier (DTC) and support vector machine (SVM), double fault measure is used to select three classifiers for further combination. MLC, DTC and MHA are selected, and their independence and diversity are evaluated. Different classifier combination strategies are experimented to extract sports field and buildings from QuickBird image. The results show that multiple classifier combination can improve the performance of image classification and target identification, and the accuracy is affected by many factors.

Keywords: multiple classifier combination, high resolution remote sensing, target identification, hierarchical classifier system, classifier selection.

1 Introduction

Target identification and extraction from remote sensing imagery is one of the most important problems in many applications. The occurrence of high spatial resolution remote sensing images provides a new way for extracting ground objects in detail. Despite their high resolution and fine description to ground objects, target identification methods from high resolution remote sensing image are still faced with such difficulties as vast data size, strong impacts of background and noises, and uncertainty in extraction process [1].

Recently, multiple classifier system (MCS) has been widely used in a variety of fields as a hot topic of pattern recognition, and multiple classifier combination or classifier ensemble has been introduced to remote sensing information processing [2-6]. In this paper, multiple classifier combination is used to target identification from high resolution remote sensing images in order to reduce the non-object noises and enhance the accuracy and reliability of target identification.

2 Multiple Classifier Combination

Multiple classifier combination can be explained briefly as deriving the final classification decision by integrating the output of multiple learning machines according to a certain combination approach. It belongs to the domain of decision level information fusion [7]. In pattern recognition and classification, the algorithm that is effective for one feature set may be unsuitable to the other feature sets, and multiple classifiers can provide the complementary information about the classified pattern on hand, so multiple classifier combination may outperform any individual classifier by integrating the advantages of various classifiers.

Usually multiple classifiers are organized by two schemes: parallel and concatenation connection [8]. In concatenation connection, outputs of the classifiers in previous level are used as the inputs into classifiers of the next level to guide next classification process. But in parallel connection, multiples classifiers are designed independently without any mutual interaction and their outputs are combined according to certain strategies, so it is quite helpful to integrate those existing individual classifiers to form a powerful recognition and classification system.

According to the output information of member classifier, classifier combination can be categorized into three levels: abstract level, rank level and measurement level [9]. For abstract level output, the output of member classifier is a class label, which means each classifier provides a label and the labels of multiple classifiers are combined further, usually majority vote is used. If the output of a classifier is the rank of one pixel belonging to every class, the combination is named as rank level combination. If the output of the member classifier can depict the quantitative degree or probability of one pixel belonging to a certain class, for example, posterior probability, and the quantitative index is used to combining multiple classifiers, it is measurement level combination.

For the target identification from high resolution remote sensing images, the scheme of parallel combination based on abstract level is used.

3 Experiments

In this experiment, multi-spectral QuickBird image (spatial resolution is 2.44m) of China University of Mining and Technology, located in Xuzhou City, Jiangsu Province, China, is used as the case study image. Training and test samples are selected by ocular interpretation and field investigation.

Multiple features are proposed to be used in high resolution image processing owing to the mutual complementation of different features [10-11]. In the experiment, multiple features including gray and spectral vector, vegetation index and texture (average, variance, entropy, correlation, second moment, contrast) are used in order to describe and extract objects effectively.

Firstly, multi-spectral image is preprocessed and useful features are extracted. The identical training sample set is used to train those member classifiers including maximum likelihood classifier (MLC), minimum distance classifier (MDC), Mahalanobis distance classifier (MHA), decision tree classifier (DTC) and support vector machine (SVM). Double fault evaluation criterion is used to select the optimal classifier combination, and this optimal combination is then used to multiple classifier system to extract the object of interest. Finally targets are recognized based on geometric feature and knowledge.

3.1 Selection of Member Classifiers

According to the land cover of study area and task of target identification, the class category consists of five classes: impervious surface, water, vegetation, sports field and shadow. Table 1 is the accuracy of all classifiers.

Table 1. The performance of individual classifiers

index	MLC	MDC	MHA	DTC	SVM
Total accuracy	81.9579	53.4680	78.8850	81.4311	82.0018
Kappa	0.7320	0.4246	0.7171	0.7443	0.7438
Accuracy rank	2	5	4	3	1

The performance of multiple classifier system is closely related with member classifiers and their combination strategy, so it is important to decide how to select classifiers from classifier ensemble and how to combine them [12-15]. In order to simplify the process, we assume that the number of classifiers selected is a fixed odd number, which is useful for majority vote combination. Here the number is assumed as 3, so there are 10 schemes when 3 classifiers are selected from a set with 5 classifiers. The double fault measure is used to assess the performance of classifier combination and the results are listed in Table 2.

Table 2. Double fault measures of different classifier combination schemes

Member classifiers	Double fault values	Member classifiers	Double fault values
1-2-3	0.0662862	1-4-5	0.129792
1-2-4	0.0727246	2-3-4	0.0632133
1-2-5	0.107112	2-3-5	0.0882353
1-3-4	0.0781387	2-4-5	0.122330
1-3-5	0.0951127	3-4-5	0.110770

Note: 1 denotes to SVM, 2 denotes MLC, 3 denotes to DTC, 4 denotes to MHA and 5 denotes to MDC.

From Table 2, it is easy to found that the combination of MLC, MHA and DTC has the smallest double fault value, so their combination is the best one and used to further target identification.

3.2 Target Identification Using Identical Feature Set

Fig.1. is the scheme of multiple classifier combination based on identical features for each member classifier [3].

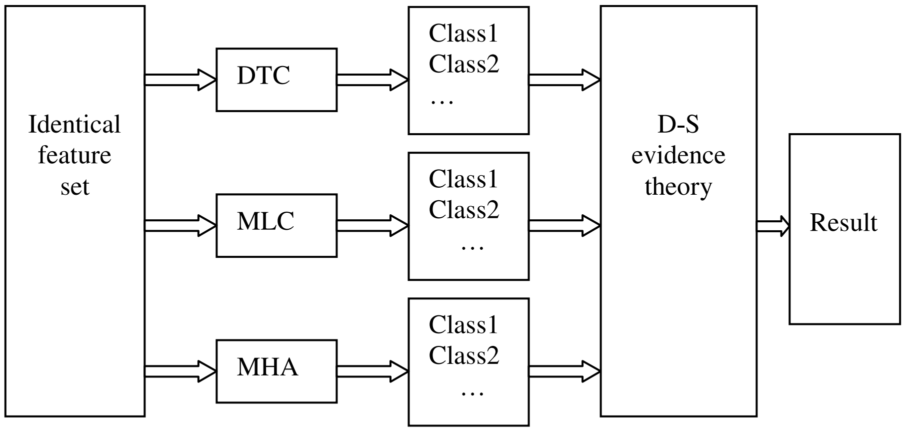


Fig. 1. The scheme of multiple classifier combination

Dempster-Shafer evidence theory is used as the fusion algorithm of outputs from multiple classifiers [13]. In multiple classifier system, the output of each classifier can be viewed as a piece of evidence, and the classification accuracy of each classifier can be used as the probability assignment function. For example, if a pixel is classified into the *i*th class (marked as C_i) by a classifier, then the basic probability assignment was assigned as: $m(C_i)=P_i$, $m(\Theta)=1-P_i$, where C_i is the *i*th class, $\Theta=\{C_1,C_2,\cdots C_{j-1},C_{j+1},\cdots C_M\}$ means the others classes and P_i is the classification accuracy of the ith class by the classifier. After finishing evidence combination, the class with maximum evidence value is used as the final decision class.

Table 3. The classification accuracy of each classifier to different classes (%)

	Impervious surface	water	vegetation	Sports field	shadow
Producer accuracy					
MLC	98.65	91.12	58.72	40.74	67.39
MHA	71.08	96.45	88.69	100.00	68.11
DTC	80.17	81.07	100.00	71.60	74.10
MCS	97.58	94.38	92.05	98.77	66.67
user accuracy					
MLC	77.27	100.00	64.43	100.00	98.25
MHA	92.57	75.64	81.23	71.68	60.43
DTC	84.59	83.54	99.09	37.54	93.35
MCS	87.21	90.88	97.41	89.89	98.23

Table 3 is the classification accuracy of each classifier to different classes when using identical input feature sets, and Table 4 is the total accuracy and kappa coefficient of individual classifier and multiple classifier system. It can be found that the combination of multiple classifiers enhance the classification accuracy to a great extent.

Table 4. Comparison of individual member classifiers with multiple classifier system

classifier	MLC	MDC	MHA	DTC	SVM	MCS
Total accuracy (%)	81.9579	53.4680	78.8850	81.4311	82.0018	90.7375
Kappa	0.7320	0.4246	0.7171	0.7443	0.7438	0.8675

3.3 Target Identification

In order to extract the target of interest, the classification results should be changed to binary image at first, and then edge tracing is conducted to the binary image, and geometric rules and prior knowledge are used to identify the targets. For example, if the target is circular building, the regions with low circular degree should be rejected. Some other geometric features include area, perimeter, rectangle degree, circle degree, central moment, centroid and so on [16].

For water and vegetation, there are not special geometric features, so the classification results are used directly. For sports field extraction, the area, perimeter and shape indicators are used. For buildings, shape index is used to the classification results. Fig.2 and Fig.3 are the result of sports field and building identification.

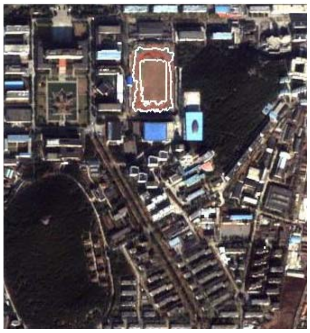


Fig. 2. Result of sports field extraction

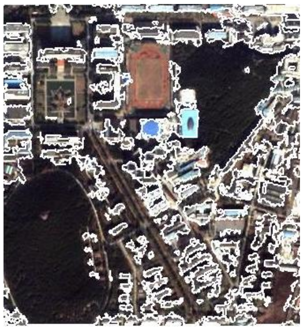


Fig. 3. Result of building identification

4 Discussions and Analysis

4.1 Quantitative Indicators for Member Classifier Selection

Member classifier selection is the prerequisite of multiple classifier system design. The simplest way is to select member classifiers based on their accuracy rank, which means those classifiers with better performance are selected. Although this idea is very straightforward and simple, it is not always the fact. The experiment of this paper has confirmed it again.

Hee-Joong Kang *et al* proposed three assessment indicators based on information theory: MC(the measure of closeness), CE(the conditional entropy), mMI(the minimization of mutual information), and they think CE is the potential clue for classifier selection, but their conclusion is not absolute yet [12]. Kuncheva *et al* analyzed the classifier selection in terms of independence of classifiers, and the conclusion is that independent classifiers can't result in higher accuracy [15].

The diversity among classifiers is necessary for multiple classifier combination. The measure of diversity can be analyzed from three levels. The first level is to use only the output of member classifier without considering the correctness of result. The second level is to consider both the output of classifier and its correctness on specific pixels. The third level is to consider not only the output of classifier and its correctness on specific pixels, but also the diversity of errors [17].

In the classifier selection in section 3.1, only one quantitative indicator is used, but many other indicators were proposed in [17]. So in this section we compare and analyze different indicators including Double-fault (DF), Kappa, Proportion of Specific Agreement(PS), Weighted count of errors and correct results(WCEC) and their impacts on classification accuracy.

The classification result based on only two classifiers can be divided into four parts: (1) samples correctly classified by both classifiers, marked as *a*; (2) samples correctly classified by the first and incorrectly by the second, marked as *b*; (3) samples incorrectly classified by the first but correctly classified by the second, marked as *c*; (4) samples incorrectly classified by both classifiers, marked as *d*. The results are illustrated in Table 7[18].

Table 5. Notation used in the dichotomous outcome for two classifiers

a	b	$p_1=a+b$
c	d	$q_1=c+d$
$p_2=a+c$	$q_2=b+d$	

Double-fault (DF) is defined as the proportion of the cases that has been misclassified by both classifiers [19].

$$DF = \frac{d}{a+b+c+d}$$

Proportion of Specific Agreement (PS) was used to measure the efficiency of classifiers combination mainly considering of samples incorrectly classified by both classifiers. The equation of Ps is:

$$PS = \frac{2d}{b+c+2d}$$

Kappa, one of the indicators which were used to measure the consistency of member classifiers, took Both of the misclassification and correctly classification into account. It could be expressed as:

$$kappa = \frac{2(ad-bc)}{p_1q_2 + p_2q_1}$$

A value of kappa below 0.40 is considered to represent poor agreement beyond chance, values between 0.40and 0.75 indicate fair agreement, and values beyond 0.75indicate excellent agreement [18].

Weighted count of errors and correct results (WCEC) takes both correct and incorrect results into consideration and gives suitable weight on them. If there are two classifiers, the Equation is

$$WCEC = a + \frac{1}{2}(b+c) - d_{different} - 5d_{same}$$

Where, $d_{different}$ stands for the number of samples incorrectly classified by both classifiers with different errors, and d_{same} represents the number of samples incorrectly classified by both classifiers, but with the same classification results.

Those indicators were pairwise measure, and for the whole combination the averaged value over all pairs of classifiers is computed. The results are illustrated in Table 6. It can be found that different indicators lead to different classifier selection scheme, so it is necessary to compare those schemes in future research.

Table 6. Comparison of diversity among classifiers

	Double fault	*kappa*	*PS*	*WCEC*	*Combination accuracy*
1-2-3	0.0663	0.2228	0.3643	0.5632	88.0597%
1-2-4	0.0727	0.2369	0.3817	0.5524	87.2695%
1-2-5	0.1071	0.2001	0.3878	0.5205	75.1975%
1-3-4	0.0781	0.2661	0.4068	0.5645	87.2695%
1-3-5	0.0951	0.1496	0.3540	0.5167	82.0896%
1-4-5	0.1298	0.2707	0.4519	0.4448	70.1932%
2-3-4	*0.0632*	0.1684	0.3277	*0.6150*	*90.7375%*
2-3-5	0.0882	*0.1023*	*0.3155*	0.5445	87.0061%
2-4-5	0.1223	0.2222	0.4129	0.4820	77.3924%
3-4-5	0.1108	0.1737	0.3807	0.4444	77.2169%
12345	0.0934	0.2013	0.3783	0.5248	89.6839%

Note: 1 is SVM, 2 is MLC, 3 is DTC, 4 is MHA and 5 is MDC

4.2 Multiple Classifier Combination Strategy

In order to analyze the impacts of classifier combination strategy on target identification accuracy, MLC, DTC and MHA is selected as member classifiers, and different combination strategies including majority vote, weighted sum, D-S evidence theory and fuzzy integral are experimented. The results are shown in Table 7. From Table 7 it can be concluded that each multiple classifier combination scheme can improve the performance of classification.

Table 7. Comparison of multiple classifier combination strategies

	SVM(best individual)	majority vote	weighted sum	D-S evidence theory	fuzzy integral
Total accuracy	82.0018	88.4987%	89.1133%	90.7375%	90.4302%
kappa	0.7438	0.8390	0.8438	0.8675	0.8624
rank		4	3	1	2

In order to compare the combination strategy, the secondary combination strategy is experimented, which means the results of different combination strategies are used as the input of the next layer combination. The flow chart is shown in Fig.4 and the accuracy statistics is listed in Table 8.

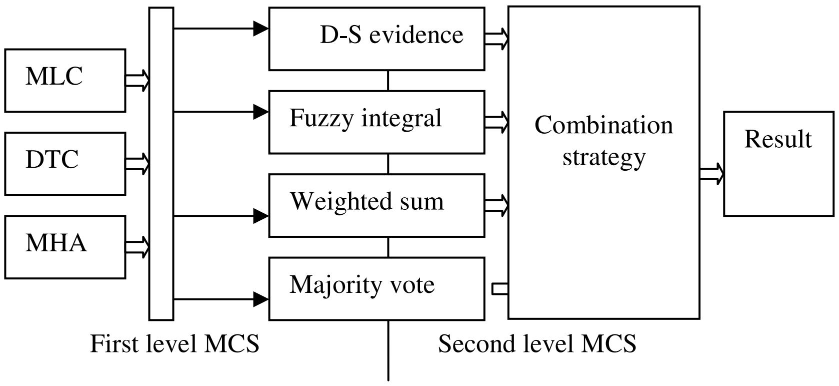


Fig. 4. Secondary combination of multiple classifier system

Table 8. Accuracy comparison of secondary classifier combination

Combination scheme	*majority vote*	*weighted sum*	*D-S evidence theory*	*fuzzy integral*
1-2-3	90.7375%	90.7375%	90.7375%	90.7375%
1-2-4	90.7375%	90.7375%	90.7375%	90.7375%
1-3-4	90.3424%	90.3424%	90.3424%	*90.8253%*
2-3-4	90.3424%	90.3424%	90.3424%	*90.8253%*
1-2-3-4	90.7375%	90.7375%	90.7375%	90.7375%

From Table 8 it can be found that the accuracy of secondary level combination is higher than that of the first level combination. Although the improvement is not strong, the performance of secondary level combination is better. There is another interesting phenomenon in Table 8, where some accuracy indicators occur repeatedly (for example, 90.7375%), and the reason for that is the strong correlativity among the inputs.

5 Conclusion

Multiple combination system is introduced to target identification from high resolution remote sensing image in this paper, and QuickBrid multi-spectral image is used to conduct a case study in the campus of China University of Mining and Technology. The whole process, including training and test sample selection, member classifier design, feature extraction, classifier selection and combination strategy determination, is investigated to classify the high resolution image and extract interested targets. Diversity of member classifiers is important to multiple classifier system, and double default is used in this paper based on the comparison to a lot of indicators. Some widely used classification combination strategies, including weighted sum, fuzzy integral, D-S evidence theory and majority vote, are experimented, and a secondary combination scheme is used to integrate the output of four combined results by the first level.

Based on the experiments and discussions in this paper, in can be concluded that multiple classifier combination can play important roles in high resolution remote sensing image classification and target identification by making full use of the abundant and detailed information in high resolution image and integrating the benefits of different classifiers. But there are still many issues for further study, for example, selection of member classifier, optimization of feature sets and determination of combination strategy, which will be emphasized in our future research.

Acknowledgments. The authors thank the support to this research from National High-Tech 863 Program of China (2007AA12Z162), the Natural Science Foundation of China (40871195), the Program for New Century Excellent Talents in University (NCET-06-0476) and the opening fund of Key Laboratory of Advanced Engineering Surveying of SBSM (ES-SBSM-(07)-01).

References

1. Shi, W.Z., Zhu, C.Q.: Road Feature Extraction from Remotely Sensed Image: Review and Prospects. Acta Geodaetica Et Cartographica Sinica 30(3), 257–361 (2001)
2. Kittler, J., Roli, F.: Multiple Classifier Systems. Springer, Heidelberg (2000)
3. Bo, Y.C., Wang, J.F.: Combining Multiple Classifiers for Thematic Classification of Remotely Sensed Data. Journal of Remote Sensing (5), 555–564 (2005)
4. Briem, G.J., Benediktsson, J.A., Sveinsson, J.R.: Multiple Classifiers Applied to Multisource Remote Sensing Data. IEEE Transactions on Geoscience and Remote Sensing 40(10), 2291–2300 (2002)
5. Mathieu, F., Chanussot, J., Benediktsson, J.A.: Decision Fusion for the Classification of Urban Remote Sensing Images. IEEE Transactions on Geoscience and Remote Sensing 44(10), 2828–2838 (2006)
6. Benediktsson, J.A., Chanussot, J., Fauvel, M.: Multiple classifiers in remote sensing: from basics to recent developments. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 501–512. Springer, Heidelberg (2007)
7. Xie, Z.X., Yu, D.R., Hu, Q.H.: Study on sensor fault tolerance with multi-classifiers of SVM fusion. Journal of Harbin Engineering University 27(supp.), 389–293 (2006)

8. Lv, Y., Shi, P.F., Zhao, Y.M.: Voting Principle for Combination of Multiple Classifiers. Journal of Shanghai Jiao Tong University 34(5), 680–684 (2000)
9. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition. IEEE Transaction on Systems, Man and Cybernetics 22(3), 418–435 (1992)
10. Lin, H., Li, J.P., Mo, D.K.: Information Identification on QuickBird Image. Journal of Image and Graphics 10(10), 1504–1511 (2005)
11. Mou, F.Y., Zhu, B.Q., He, H.Z.: Multiple Features Based Analysis of High-resolution Remotely Sensed Imager. Bulletin of Surveying and Mapping 10, 4–7 (2004)
12. Kang, H.J., Doermann, D.: Selection of Classifiers for the Construction of Multiple Classifier Systems. In: Proceedings of the Eight International Conference on Document Analysis and Recognition, pp. 263–268 (2005)
13. Liu, C.P., Dai, J.F., Zhong, W., et al.: Multi-source Remote Sensing Information Based on Fuzzy Evidence Theory. Pattern Recognition and Artificial Intelligence 6, 213–218 (2003)
14. Congahon, R.G., Green, K.: Assessing the Accuracy of Remotely Sensed Data: Principles and Practices. Lewis Publishers (1999)
15. Kuncheva, L.I., Whitaker, C.J., Shipp, C.A.: Is Independence Good for Combining Classifiers? In: Proceedings of ICPR 2000, vol. 2, pp. 168–171 (2000)
16. Inglada, J.: Automatic recognition of man-made objects in high resolution optical remote sensing images by SVM classification of geometric image features. ISPRS journal of photogrammetry and remote sensing 62, 236–248 (2007)
17. Matti, A., Jorma, L.: Using diversity of errors for selecting members of a committee classifier. Pattern Recognition 39, 608–623 (2006)
18. Petrakos, M., Benediktsson, J.A.: The Effect of Classifier Agreement on the Accuracy of the Combined Classifier in Decision Level Fusion. IEEE Transactions on Geoscience and Remote Sensing 39(11), 2539–2545 (2001)
19. Giancinto, G., Roli, F.: Design of Effective Neural Network Ensembles for Image Classification Purposes. Image Vision Computer (19), 697–705 (2001)

Neural Network Optimization for Combinations in Identification Systems

Sergey Tulyakov and Venu Govindaraju

Center for Unified Biometrics and Sensors,
University at Buffalo, Buffalo, USA
{tulyakov,venu}@cubs.buffalo.edu
http://www.cubs.buffalo.edu/

Abstract. In this paper we investigate the construction of combination functions in identification systems. In contrast to verification systems, the optimal combination functions for identification systems are not known. In this paper we represent the combination function by means of a neural network and explore different methods of its training, so that the identification system performance is optimized. The modifications are based on the principle of utilizing best impostors from each training identification trial. The experiments are performed on score sets of biometric matchers and handwritten word recognizers. The proposed combination methods are able to outperform the likelihood ratio, which is optimal combination method for verification system, as well as, weighted sum combination method optimized for best performance in identification systems.

Keywords: Classifier combination, identification system, biometric matchers, neural network.

1 Introduction

Suppose we have a matching system with N registered or enrolled classes. Given some input, the system has to find its best match to any of the N enrolled classes. We call such matching system an *identification system*. In contrast to more general N-class pattern classification problem setup, we imply that the matching score of the input to the enrolled class is derived using only input and enrolled templates. Consequently, identification systems can deal with variable and large number of classes N and no retraining of the matching algorithm is needed. The examples of identification systems include biometric identification systems and handwritten word recognition; both can contain large and variable number of classes, persons or lexicon words, and matching algorithms, as a rule, calculate the matching score using only two, input and enrolled, templates.

Identification systems are different from verification systems. In *verification systems* the possible class of the input is provided beforehand; the system only performs the match of the input to the enrolled template of the specified class, and depending on the matching score outputs accept or reject decision. In *identification system* we have to match the input against all enrolled templates and output the class matching the input best. For evaluating performance of verification systems we can use ROC curves, and for evaluating performance of identification systems we can calculate the correct identification rate or use rank measures such as cumulative match curves (CMC).

In this paper we consider the combinations of matching scores in identification systems. Though we used only pairs of matchers for combination, the presented algorithms can be applied to situations with few matchers. Each matcher $j = 1, 2$ produces sets of matching scores s_i^j assigned to each of $i = 1, \ldots, N$ classes. A combination function f is used to combine 2 matching scores corresponding to each class: $S_i = f(s_i^1, s_i^2)$. In identification systems the classification result C is determined as

$$C = \arg \max_{i=1,\ldots,N} f(s_i^1, s_i^2) \tag{1}$$

In verification systems, on the other hand, we compare the value of combination function to some threshold and make accept/reject decision.

The optimal combination function for verification system is well-known [1,2]; such function should have decision surfaces separating two types of score pairs (s_i^1, s_i^2), genuine and impostor. Optimal Bayes classifier separating genuine and impostor score pairs is obtained by the ratio of genuine and impostor score densities (*likelihood ratio*):

$$f_{lr}(s_i^1, s_i^2) = \frac{p_{gen}(s_i^1, s_i^2)}{p_{imp}(s_i^1, s_i^2)} \tag{2}$$

and can be well approximated if the number of matchers (two in our case) is relatively small.

On the other hand, the solution to finding optimal combination function for identification system is not yet known. As we showed in [2], likelihood ratio f_{lr} is optimal if matching scores assigned to different classes are statistically independent. If they are dependent, likelihood ratio might be non-optimal, and the performance of combined system can be even worse than the performance of a single matcher. The scores assigned to different classes are usually dependent since they are derived using same input template.

The goal of this paper is to investigate the approaches of constructing combination function for identification systems with the help of multi-layer perceptron. In particular, we present two methods of training neural network resulting in increased identification system performance.

2 Previous Work

Previous work in classifier combinations and combinations of biometric matchers makes little distinction on whether the considered system is verification or identification system. For example, Kittler et al. [3] derive the combination rules assuming identification system, but test them using verification system. Besides, the independence of matching scores assigned to different classes is assumed in that work.

As another example, Lee et al. [4] explicitly reduce the problem of combining matchers in a biometric identification system to the task of applying a classifier (SVM) trained for an equivalent verification system. Since the combination function is trained for verification system, it may not produce an optimal combination algorithm for identification systems.

Whereas most research in combinations of biometric matchers deals with verification systems, e.g. [5], the earlier research in classifier combinations dealt with more general classifiers. Effectively, many of the earlier approaches were learning combination functions of the following type:

$$C = \arg \max_{i=1,\dots,N} f_i(\{s_k^1\}_{k=1,\dots,N}, \{s_k^2\}_{k=1,\dots,N}) \quad (3)$$

instead of less complex combinations of Eq. 1. For example, Bayesian and Dempster-Shafer combination methods of [6] require learning confusion matrices for each classifier participating in the combination. The Behavior-Knowledge Space combination method of [7] requires learning a decision space of a set of classifiers participating in the combination. Although these approaches can be considered to be somewhat optimal, they could be applied only in situations with a small number of classes. However, in our applications of biometrics and handwritten word recognition, the number of classes N is of the order of thousands and we are forced to construct combinations of the Eq. 1 type.

Some of the earlier works on the training of pattern recognition systems recognized the need to train the algorithms with the goal of minimizing the classification errors. As noted in [8], the traditional neural network training involving MSE (mean squared error) minimization might not result in the neural network having minimum classification errors. Different methods of training neural networks for classification error minimization have been proposed [8,9,10]. Though our application of neural network used as combination function of Eq. 1 is different from the application of neural network as pattern classifiers in these previous works, which are rather similar to Eq. 3, we employ a training principle similar to principle proposed in those works - we will be utilizing training samples proved to be most difficult for classification.

We have previously underscored the need to have a separate training procedures of combination algorithms in verification and identification systems [2]. Furthermore, we proposed some heuristic methods of constructing combination functions for identification systems in [11]. In this paper we are looking for the ways to change the training procedures of traditional multilayer perceptron neural networks, so that the resulting combination function has optimized performance in identification systems. Methods considered in this paper can be viewed as a more automated and generalized compared to the heuristic methods described in [11].

3 Optimizing Combination Functions for Identification Systems

3.1 Weighted Sum Combination

One of the most frequently used methods for combining matching scores in identification systems is the weighted sum rule. In our case, we combine only two matchers and the weighted sum combination function can be written as

$$f(s^1, s^2) = ws^1 + (1 - w)s^2 \tag{4}$$

The weight w can be chosen heuristically so that the better performing matchers have a bigger weight [5]. The optimal weights can be also estimated for linear combinations of classifiers subject to the minimization of classification error [12].

In our experiments we have trained the weights so that the number of successful identification trials on the training set is maximized. The previously proposed methods of training resulting in the minimization of classification error [12] are not directly applicable due to much bigger number of classes in our case. Since we have only two matchers in all our configurations, it was possible to utilize a brute-force approach: we calculate the correct identification rate of the combination function $f(s^1, s^2) = ws^1 + (1 - w)s^2$ for different values of $w \in [0, 1]$, and find w corresponding to the highest recognition rate. Despite being brute-force, due to simplicity of weighted sum method, this approach was the fastest to train.

3.2 Minimizing Classification Error and Iterative Learning

If we perform training for verification system, we can treat genuine and impostor scores from different identification trials separately. Indeed, the optimal combination in the form of likelihood ratio (Eq. 2) uses separately approximated genuine and impostor densities, and any other algorithm can do the same. But in order to perform training of combination function for identification system, we have to consider the scores in each identification trial as a single training sample, and train the combination function on these samples. This is precisely the technique used to train the weighted sum rule for identification systems (Section 3.1). For each training identification trial we check whether the genuine score pair produced greater combined scores than all the impostor score pairs. By counting the numbers of successful trials we were able to choose the proper weights. Although the weighted sum rule provides a reasonable performance in our applications, its decision surfaces are linear and might not completely separate the generally non-linear score distributions. Therefore we explore more complex combination functions trained with the available training set.

In this paper we explore the approximation of combination function $f(s^1, s^2)$ by means of neural network, multilayer perceptron. Although the previous work in neural network optimization for minimizing misclassification errors was in constructing classifiers and not their combinations [8,9,10], we apply similar optimization criteria for the training. In [9] several optimization criteria were explored. The general solution consists in constructing a smooth misclassification cost function giving different weights to different errors of currently trained neural network, and modifying neural network by gradient descent method to reduce the cost. In our case, we used one particular case of such cost - the cost incurred by the largest possible error from the best impostor. Such cost is an extreme case of parametric cost functions considered in [9], where the parameter is chosen so that the cost function uses only best impostor.

Another difference of our approach with previous research in minimum classifier error optimization of neural networks is that during network update the input to our neural network can consist of only one score pair (this is difference between Eq. 3 and Eq. 1). As a consequence, we are not only required to modify the cost function determined by network outputs, but we also need to provide a proper score pair as the training input. By considering only the best impostor score pair we are able to do it.

In order to implement our algorithms, we need to be able to determine what is the best impostor score pair in each training identification trial. The best impostor depends on the currently trained combination function. Therefore, for our methods we use the following iterative training procedure:

1. Make initialization of $f(s^1, s^2)$.
2. For each training identification trial find the impostor score pair with the biggest value of the combined score according to currently trained $f(s^1, s^2)$.
3. Update $f(s^1, s^2)$ by using genuine score pair and found best impostor score pair of one identification trial.
4. Repeat steps 2-3 for all training identification trials.
5. Repeat steps 2-4 for predetermined number of training epochs.

Note, that proposed training procedure based on best impostors does not explicitly model the dependence between matching scores assigned to different classes. Though such modeling can be very helpful for improving the performance of combination algorithms, it leads to a different type of combinations not defined by Eq. 1 [13]. In current paper we are interested in modified optimization criteria (minimizing classification error), and by using this criteria we implicitly account for dependencies between matching scores assigned to different classes.

3.3 Neural Network Training for Identification Systems

Neural networks, especially multilayer perceptrons, allow approximation of arbitrary functions. Therefore, it should be possible to train a neural network to represent the optimal combination function in identification systems. This approach can be viewed as a generalization of the combination method using a sum of logistic functions described in our previous work [11]. A neural network with logistic activation functions and a single layer of hidden nodes directly corresponds to the combination function consisting of the sum of logistic functions. By utilizing more than one hidden layer and by using generic training of the neural network, it is possible to obtain better combination function than by using ad hoc structure and training procedure of the sum of logistic functions [11].

We compare three approaches for training a neural network for the combination task at hand. The first approach is the traditional training using separate genuine and impostor scores. The other two approaches focus on minimizing the misclassification rate, and the genuine and impostor scores are not treated separately.

1. Traditional training: random impostor score pairs are used alongside with genuine score pairs.

2. Best impostor training: following iterative training procedure, the best impostor score pair is found from an identification trial and used together with genuine score pair to update neural network.
3. Mixed scores training: we use best impostor from the identification trial for training only if there is a failure in one combined classifier. More precisely, let (s^1_{gen}, s^2_{gen}) and (s^1_{bi}, s^2_{bi}) denote the genuine and best impostor score pairs for the current identification trial. We update the neural network only if $s^1_{gen} > s^1_{bi}$ and $s^2_{gen} < s^2_{bi}$) or ($s^1_{gen} < s^1_{bi}$ and $s^2_{gen} > s^2_{bi}$). This training method can be viewed as a combination of best impostor neural network training and the conditional training of the sum of logistic functions combination method [11].

Our goal is to train the neural network so that the misclassification rate is minimized. As we discussed in section 3.2, the configuration of our neural network is different from the networks trained with the purpose of classifier error minimization. But we can notice that used optimization criteria are similar. Indeed, by considering the best impostor we effectively use the extreme case of parametric cost functions presented in [9]. During our training of neural network we employ the mean square error defined for genuine and best impostor samples; such choice of error calculation corresponds to considering the square polynomial functions for cost calculations in [9]. The mixed score training implies additional selection of training samples, and can also be represented by a proper choice of cost function family.

4 Experiments

4.1 Handwritten Word Recognizers

We consider the application of handwritten word recognizers in the automatic processing of United Kingdom mail. The destination information of the mail piece contains the name of the postal town or county. After automatic segmentation of the mail piece image, the goal of the handwritten word recognizer is to match the hypothesized town or county word image against a lexicon of possible names, which contains 1681 entries.

We use two handwritten word recognizers for this application: Character Model Recognizer (CMR)[14] and Word Model Recognizer (WMR)[15]. Both recognizers employ similar approaches to word recognition: they oversegment the word images, match the combinations of segments to characters and derive a final matching score for each lexicon word as a function of the character matching scores.

Our data consists of three sets of word images of approximately the same quality. The data was initially provided as these three subsets and therefore we did not regroup them. The images were manually truthed and only those images containing any of the 1681 lexicon words were retained. The word recognizers were run on these images and their match scores for all 1681 lexicon words were saved. Note, that both recognizers reject some lexicon entries if, for example, the lexicon word is too short or too lengthy for the presented image. We assume that in real systems such rejects will be dealt with separately (it is possible that the lexicon word corresponding to image truth will be rejected), but for our combination experiments we keep only the scores of those lexicon words which are not rejected by either of the recognizers. Thus for each image I_k we have a variable number N_k of score pairs (s_i^{cmr}, s_i^{wmr}), $i = 1, \dots, N_k$ corresponding to non-rejected lexicon words. One of these pairs corresponds to the true word of the image which we refer to as 'genuine' scores, and the other 'impostor' score pairs correspond to non-truth words.

After discarding images with non-lexicon words, and images where the truth word was rejected by either recognizer, we are left with three sets of 2654, 1723 and 1770 images and related sets of score pairs. We will refer to the attempt of recognizing a word image as an identification trial. Thus each identification trial has a set of score pairs (s_i^{cmr}, s_i^{wmr}), $i = 1, \dots, N_k$ with one genuine score pair and $N_k - 1$ impostor pairs. The scores of each recognizer were also linearly normalized so that each score is in the interval $[0, 1]$ and bigger score implies a better match.

Since our data was already separated into three subsets, we used this structure for producing the training and testing sets. Each experiment was repeated three times. Each time one subset is used as a training set, and the other two sets are used as test sets. The final results are derived as averages of these three training/testing phases.

4.2 Biometric Person Matchers

We used biometric matching score set BSSR1 distributed by NIST[16]. This set contains matching scores for a fingerprint matcher and two face matchers 'C' and 'G'. Fingerprint matching scores are given for left index 'li' finger matches and right index 'ri' finger matches. For our experiments we used four combinations involving both fingerprint and face score subsets: 'li&C', 'li&G', 'ri&C' and 'ri&G'

Though the BSSR1 score set has a subset of scores obtained from the same physical individuals, this subset is rather small - 517 identification trials with 517 enrolled persons. Therefore we used larger subsets of fingerprint and face matching scores of BSSR1 by creating virtual persons. The fingerprint scores of a virtual person come from a physical person and the face scores come from a different individual. The scores are not reused, and thus we are limited to a maximum of 6000 identification trials and a maximum of 3000 classes (or enrolled persons). Some enrollees and some identification trials also needed to be discarded since the corresponding matching scores were invalid probably due to enrollment errors. Finally, we split the data into two parts - 2991 identification trials with 2997 enrolled persons, with each part used as training and testing sets in two phases. The final results are the averages of these two phases.

4.3 Experimental Results

In the likelihood ratio method we reconstructed the densities using the Parzen window method with Gaussian kernels. The window widths are found by maximum likelihood leave-one-out cross validation method on a training set. Note that the reconstructed densities $p_{gen}(s^1, s^2)$ and $p_{imp}(s^1, s^2)$ of the likelihood ratio combination function 2 are two-dimensional. Given a large number of training samples, using two-dimensional kernels in the Parzen method results in a good approximation of the densities [17].

For the weighted sum combination method 4, as well for other methods, we use separate training and testing subsets. It is worth noting, that despite of only single weight w to be found, weighted sum method indeed has a slightly lower performance on the testing sets than on the training set.

In all the cases of the neural network methods we have the same configuration - multilayer perceptron with configuration 2-8-9-1, sigmoid activation functions and backpropagation training. We keep the default parameter settings of the neural network library [18]. About 100 training epochs are required to get the best performance with minimal overfitting effect. Since we did not have a separate validation set for the biometric dataset, we decided to run the training for 300 epochs and choose the best performance numbers on test datasets.

Table 1. The results of experiments. Numbers represent the correct identification rates (in %).

Matchers	Likelihood Ratio	Weighted Sum Rule	Traditional Training	Best Impostor Training	Mixed Scores Training
CMR&WMR	69.84	81.58	76.69	80.54	**81.67**
li&C	97.24	97.23	97.01	97.26	**97.39**
li&G	95.90	95.47	96.00	96.07	**96.29**
ri&C	98.23	98.09	98.21	98.26	**98.33**
ri&G	97.14	96.82	97.41	**97.43**	97.38

The results of the experiments are presented in Table 1. The numbers in the table refer to the correct identification rates, that is the percentage of trials in which the genuine score receives the best score compared to impostor scores. In general, neural networks showed slightly better results which can be explained by their superior trainability compared to density based likelihood ratio method and linear weighted sum method.

As we discussed in [2], the likelihood ratio method actually fails for combination of word recognizers - it has lower performance than WMR alone. Such result is explained by the strong dependence between WMR's matching scores assigned to different classes. But likelihood ratio has slightly better performance than weighted sum for combination of biometric matchers due to weaker dependence between scores and the inability of weighted sum to model non-linear decision boundaries. The goal of considered neural network combination is to be able to outperform both likelihood ratio and weighted sum. As we can see from Table 1, our modifications to neural network training achieved this task. The last modification, mixed scores training, is able to outperform the likelihood ratio weighted sum combination in all cases.

5 Summary

Verification and identification systems possess different optimal combination functions, and therefore require different training procedures. The optimal combination function for verification systems coincides with the likelihood ratio of genuine and impostors scores. We can approximate this function directly by reconstructing score densities, as we did in this paper, or use traditional pattern classification algorithms trained to separate genuine and impostor scores. The optimal combination function for identification system, on the other hand, is difficult to find.

In this paper we investigated the approaches of constructing combination functions for identification systems by means of neural networks. Previous works in neural network optimizations suggest the possibility that our optimization modifications might have a property of optimality. The experiments on biometric matchers and handwritten word recognizers show that proposed methods are able to outperform likelihood ratio, as well as traditionally used in identification system, weighted sum combination method.

References

1. Prabhakar, S., Jain, A.K.: Decision-level fusion in fingerprint verification. Pattern Recognition 35(4), 861–874 (2002)
2. Tulyakov, S., Govindaraju, V., Wu, C.: Optimal classifier combination rules for verification and identification systems. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 387–396. Springer, Heidelberg (2007)
3. Kittler, J., Hatef, M., Duin, R., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence, 226–239 (March 1998)
4. Lee, Y., Lee, K., Jee, H., Gil, Y., Choi, W., Ahn, D., Pan, S.: Fusion for multimodal biometric identification. In: Kanade, T., Jain, A., Ratha, N.K. (eds.) AVBPA 2005. LNCS, vol. 3546, pp. 1071–1079. Springer, Heidelberg (2005)
5. Snelick, R., Uludag, U., Mink, A., Indovina, M., Jain, A.: Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(3), 450–455 (2005)
6. Xu, L., Krzyzak, A., Suen, C.Y.: Methods for combining multiple classifiers and their applications to handwriting recognition. IEEE transactions on System, Man, and Cybernetics 23(3), 418–435 (1992)
7. Huang, Y., Suen, C.: A method of combining multiple experts for the recognition of unconstrained handwritten numerals. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(1), 90–94 (1995)
8. Hampshire II, J.B., Waibel, A.H.: A novel objective function for improved phoneme recognition using time-delay neural networks. IEEE Transactions on Neural Networks 1(2), 216–228 (1990)
9. Juang, B.H., Katagiri, S.: Discriminative learning for minimum error classification [pattern recognition]. IEEE Transactions on Signal Processing, IEEE Transactions on see also Acoustics, Speech, and Signal Processing 40(12), 3043–3054 (1992)
10. Nedeljkovic, V.: A novel multilayer neural networks training algorithm that minimizes the probability of classification error. IEEE Transactions on Neural Networks 4(4), 650–659 (1993)

11. Tulyakov, S., Wu, C., Govindaraju, V.: Iterative methods for searching optimal classifier combination function. In: First IEEE International Conference on Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007, pp. 1–5 (2007)
12. Ueda, N.: Optimal linear combination of neural networks for improving classification performance. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(2), 207–215 (2000)
13. Tulyakov, S., Govindaraju, V.: Use of identification trial statistics for combination of biometric matchers. IEEE Transactions on Information Forensics and Security 3(4), 719–733 (2008)
14. Favata, J.: Character model word recognition. In: Fifth International Workshop on Frontiers in Handwriting Recognition, Essex, England, pp. 437–440 (1996)
15. Kim, G., Govindaraju, V.: A lexicon driven approach to handwritten word recognition for real-time applications. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(4), 366–379 (1997)
16. Nist biometric scores set, http://www.nist.gov/biometricscores/
17. Tulyakov, S., Govindaraju, V.: Utilizing independence of multimodal biometric matchers. In: Gunsel, B., Jain, A.K., Tekalp, A.M., Sankur, B. (eds.) MRCS 2006. LNCS, vol. 4105, pp. 34–41. Springer, Heidelberg (2006)
18. Nissen, S.: Implementation of a fast artificial neural network library (fann). Technical report, Department of Computer Science, University of Copenhagen (2003)

MLP, Gaussian Processes and Negative Correlation Learning for Time Series Prediction

Waleed M. Azmy[1], Neamat El Gayar[1,2], Amir F. Atiya[3], and Hisham El-Shishiny[4]

[1] Faculty of Computers and Information, Cairo University, 12613 Giza, Egypt
{w.azmy,n.elgayar}@fci-cu.edu.eg
[2] Center of Informatics Science, Nile University, Giza, Egypt
nelgayar@nileuniversity.edu.eg
[3] Faculty of Engineering, Cairo University, Giza, Egypt
amir@alumni.caltech.edu
[4] IBM Center of Advanced Studies in Cairo, IBM Cairo Technology Development Center, Giza, Egypt
shishiny@eg.ibm.com

Abstract. Time series forecasting is a challenging problem, that has a wide variety of application domains such as in engineering, environment, finance and others. When confronted with a time series forecasting application, typically a number of different forecasting models are tested and the best one is considered. Alternatively, instead of choosing the single best method, a wiser action could be to choose a group of the best models and then to combine their forecasts. In this study we propose a combined model consisting of Multi-layer perceptron (MLP), Gaussian Processes Regression (GPR) and a Negative Correlation Learning (NCL) model. The MLP and the GPR were the top performers in a previous large scale comparative study. On the other hand, NCL suggests an alternative way for building accurate and diverse ensembles. No studies have reported on the performance of the NCL in time series prediction. In this work we test the efficiency of NCL in predicting time series data. Results on two real data sets show that the NCL is a good candidate model for forecasting time series. In addition, the study also shows that the combined MLP/GPR/NCL model outperforms all models under consideration.

Keywords: Time series prediction, Negative Correlation Learning, MLP, Gaussian Processes, NN3 data, diversity, Wilcoxon.

1 Introduction

Time series forecasting is a type of problem whereby the temporal structure and ordering of the data is utilized in some way [1]. To forecast a time series, a wide variety of approaches are available. Linear models estimate the unknown value as a linear combination of other values. This class of models includes AR, ARMA, ARX, ARMAX, Box-Jennkins, etc [1]. In spite of being quite simple to use, well recognized and do not suffer too much from the choice of the structural parameters; simple linear model predictions can obtain poor and useless results when dealing with a number of real world situations. Nonlinear prediction models have been used to overcome this limitation including machine learning techniques like Artificial Neural Network (ANN), Simple Bayesian Regressors, K-Nearest Neighbour (K-NN), Bayesian Neural Network (BNN), fuzzy models and Hidden Markov Models [2] [3] [4] [5] [6] [7]. Despite their more general character, and therefore their extended potential, nonlinear tools suffer from high complexity and the setting of many user-controlled structural parameters that very often can rely on the users experience and can require high computation time and resources to be adjusted for best performance.

In a recent study [8] a large scale empirical comparison of eight different computational intelligence models for time series prediction was conducted. The study revealed that the standard multilayer perceptron neural network (MLP)and the Gaussian process regression (GPR)are the best models to consider. The study also presented a thorough methodology for data pre-processing techniques and model parameter setting.

A common difficulty to all methods of time series prediction is choosing the appropriate model for the application considered. The forecast procedure is usually affected by many factors and hence alternative approaches can capture this variability. Combining several methods rather than attempting to select the best one can reduce errors arising from faulty assumptions, bias or mistakes in the data.

Ensemble methods have been shown both theoretically and empirically to outperform single predictors on a wide range of tasks [9]. Neural network ensembles are well accepted as a way to combine a group of weak models in order to make a composite, stronger one. It has been shown that low correlation of error and diverse members may give better ensemble performance [10].

Many techniques for creating diverse ensemble members are built using heuristics and intuition. Some attempts to achieve a good compromise between accuracy and diversity include variants of *bagging, boosting and stacking* techniques [11] [12] [13].

In a further study a combined MLP/GPR model was sucessful in the prediction of a large collection of time series data. The model combination used included some aspects of forecast combination using the simple average and model selection based on the training set and the validation set performance. This model was among the winning models in two major international forecasting competition the *NN3 and NN5 Time series Forecasting Competitions* [14] [15].

On the other hand the negative correlation learning (NCL) algorithm suggests an alternative way creating diverse predictors. NCL algorithm has attempted to explicitly quantify diversity, and incorporate it into a learning algorithm [16]. NCL has been used in a variety of applications such as (classification, regressions, incremental learning algorithms, bioinformatics ... etc) and proved to be effective as reported in [17] [18] [19].To our knowledge, there were no reported attempts to test and compare the performance of NCL against other machine learning methods when applied to time series prediction.

The objective of this study is to test the performance of the NCL algorithm when applied to time series prediction and to compare it to the MLP and the GPR models. The other goal is to apply the considered methodology to the tourist arrivals forecasting problem. This is one of the important applications of time series forecasting, and it has attracted a lot of research interest. The need for accurate tourism forecasting is of particular importance for better planning and business decisions due to the significant contribution of the tourism industry to the economies of many countries. In this paper we consider inbound tourism to Egypt from 36 source countries. As such, these time series could be a useful benchmark for comparing the considered models. The other benchmark that we used is the NN3 competition time series data. These are essentially business-type time series that possess some similarities with the time series data obtained from the tourism domain in terms of nonlinearity, trend and seasonality. Results show that NCL offers an improved accuracy compared to the MLP and the GPR models. In this work we also devise a new combined model that was found most efficient when used to forecast time series data. To draw final conclusions we use the *Wilcoxon signed-ranks* test to verify the significance of the obtained results.

The paper is organized as follows; Section 2 briefly describes the MLP, the GPR models and the NCL algorithm. Section 3 outlines methods for data pre-processing and model parameter optimization. Section 4, describes the used data sets, and explains the details of the experiments conducted. In Section 5 results are presented and discussed. Finally, the paper is concluded in Section 6.

2 Machine Learning Models for Time Series Forecasting

In this section we briefly introduce two machine learning models who have been proven to be successful for time series prediction; the MLP and the GPR. We also briefly review the NCL algorithm.

2.1 Multi-layer Perceptron

The multilayer perceptron is perhaps the most popular network architecture in use today both for classification and regression. The MLP consists of several layers. The MLP output is given by: $\hat{y} = v_0 + \sum_{j=1}^{NH} v_j g(\omega_j^T \grave{x})$; where $\grave{x}$ is the input vector x augmented with 1, i.e. $\grave{x} = (1, x^T)^T$, ω_i is the weight vector for j^{th} hidden node, $v^0, v^1, v^2 ... v^{NH}$ are the weights for the output node, and $\hat{y}$ is the network output. The function g represents the hidden node output, and it is given in terms of a squashing function, in our study for example we use the logistic function: $g(u) = \frac{1}{1+\exp(-u)}$.

The MLP is a heavily parameterized model, and by selecting the number of hidden nodes NH we can control the complexity of the model. The most interesting property of the MLP is the capability to work as a universal approximator [20].

2.2 Gaussian Processes

Gaussian process regression (GPR) has many desirable properties, such as ease of obtaining and expressing uncertainty in predictions, the ability to capture a wide variety of behavior through a simple parameterization, and a natural Bayesian interpretation. Because of this GPR has been suggested as replacements for supervised neural networks in non-linear regression, and have been extended to handle classification tasks [21].The ability in GP to fit arbitrary functions or surfaces is its nonparametric flexibility.

Gaussian process regression models are constructed from classical statistical models by replacing latent functions of parametric form (e.g. linear functions, truncated Fourier or Wavelet expansions, multi-layer perceptrons) by random processes with Gaussian prior. The posterior distribution of a to-be-predicted function value can then be obtained using the assumed prior distribution -normal distribution- by applying simple probability manipulations [8]. (See detailed illustration of the GP model in Rasmussen and Williams [22]).

2.3 Negative Correlation Learning Algorithm

Generally in building an ensemble, one of the elements required for accurate prediction is recognized to be error "diversity". NCL algorithm suggests an alternative way for managing this diversity by updating the backpropagation's delta rule in training the predictors in the ensemble.In constructing the base set of learners, it is well appreciated that the individuals should exhibit different patterns of generalization.

NC learning works with a penalty term attached to the normal MSE error function. A coefficient can be used to vary the emphasis on the penalty term. With a coefficient of zero, the NC learning algorithm is exactly equivalent to a simple ensemble of learners. With a higher coefficient - which means more emphasis on diversity significantly faster convergence and lower generalization error is observed on a number of problems [18]. (see [23] for a detailed description of the NC training algorithmn). In our experiments we choose the learning rate parameter α to be 0.1. The other parameters of the NCL are determined by crossvalidation as will be outlined in the next sections.

3 Data Preprocessing and Model Parameter Setting

In this section we describe our methodology for preprocessing the time series data used and how model parameters selection is performed.

In general we adopt the following methodology in our experiments. First we conduct a pre-processing step on the time series data. Then we perform a parameter optimization step for the model used. We use the model with the optimized parameters to learn and finally test it using a cross validation approach. Usually a post-processing stage is needed before the final testing to recover the original time-series data. In what follows we provide more details.

Preprocessing the time series data is a necessary step as it can considerably impact the performance of the prediction model.(See [8] for a more thorough discussion). Particularly, time series data may include patterns of seasonality which have to be dealt with before applying the prediction model. Therefore a seasonal test is carried out to detect the presence of seasonality. If detected, the time series is deseasonalized. Also, a log transformation is performed to limit the effects of exponential growth of some time series. As is usually performed for neural networks, we scale time series to be in the range [1.1]. In summary, we perform the following transformation by the following order: 1. Log transformation., 2. Deseasonalization -if needed and 3. Scaling.

On the other hand, the used models require some key parameters that need to be determined. We use K-fold validation for the purpose of determining these parameters. In this approach we have a set of M examples and partition them into K sets ("folds") of size M/K. For each fold, the model is trained on the other folds and then tested on this fold. The total testing error is then used to select the best parameter values.

The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing.

The time series is post processed in the reverse order and the parameter of the linear scaled time series (slope, intercept) are used to scale the time series to its actual values. If the time series has a seasonal pattern and was deseasonalized, then we add the seasonal factors.

4 Data and Experiments

In this study we use two data sets to test the models under consideration. In particular we use the "NN3 data set" and data describing the monthly tourism arrivals to Egypt by nationality.

From the NN3 competition data sets we use the time series of length greater than 50 points in order to have enough data for neural network ensemble training. This reduced the number of time series to 61 time series. The NN3 data set consists of monthly time series data with different properties, seasonal patterns and noise effects. (For more details see [14]).

On the other hand, we considered the problem of forecasting inbound tourism demand for Egypt. Specifically, we consider two type of time series for 36 major source countries: a time series describing the monthly tourist arrivals from origin country and time series indicating the number of nights spent by tourists from origin country. In addition for each type of time series the aggregate over all counties was used. Therfore in total the data was composed of $((36*2)+2) = 74$ time series spanning the period from 1993 to 2007.

For the NCL model, we need to optimize the number of lags, penalty factor λ and number of hidden nodes. We vary the number of lags between [1,6] to find the optimum one, penalty factor λ is optimized between [0, 1] with step equals to 0.1 and the number of hidden nodes is varied between [1, 12] with step equals to 1. (see [8] [14] for the details of the parameter determination for MLP and GPR).

We use the holdout method to evaluate the accuracy of the predictive model. The hold out data is the test data set. The over all accuracy is take from only one run.

5 Results and Discussion

We use the *Symmetric Mean Absolute Percentage Error*, $SMAPE$ as the error measure, as described by equation (2) (y_m is the target output and $\hat{y_m}$ is the prediction). $SMAPE$ has several advantages including removing scale and ease of interpretation. Table 1, presents the results of the tested models on the NN3 data and the tourism data. The results are calculated as the average $SMAPE$ of all time series in each data set. In the NN3 data set, we used the validation period of 18 data points following the instructions of the competition to use the last 18 points to validate the goodness of fit, but we did not use them to evaluate and rank the models performance. On the other hand, we used 6 point forecast horizon for the tourism data set.

Table 1 compares the error percentage of the MLP, GPR, combined MLP-GPR (MLP/GPR), the NCL and finally the combined (MLP/GPR/NCL) model. The combination we used is based on the average rule: $f = \frac{1}{N}\sum_{i=1}^{N} f_i$; where N is the number of combined models and f_i is the forcast obtained by model i. There is impirical evidence that equal weights and simple combination schemes do better than more sophisticated rules[20].

The results show that for both data sets the combined MLP/GPR/NCL model achieves the best performance. As follows we introduce the Wilcoxon signed-ranks test and use it to compute the statistical significance of the obtained results.

$$SMAPE = \frac{1}{M}\sum_{m=1}^{M}\frac{|\hat{y}_m - y_m|}{(|\hat{y}_m| + |y_m|)/2} \tag{1}$$

5.1 The Wilcoxon Signed-Ranks Test

Over the last years, the machine learning community has become increasingly aware of the need for statistical validation of the published results. Statistical

Table 1. The average $SMAPE$ (%) for each model in both data sets

Data Sets	NN3	Tourism dataset
MLP	16.98	24.86
GPR	16.15	25.67
MLP/GPR Ensemble	15.50	24.52
NCL	15.06	24.55
NCL/MLP/GPR Ensemble	**13.65**	**23.76**

evaluation of experimental results has been considered an essential part of validation of new machine learning methods for quite some time [24].The tests used have however long been rather naive and unverified. The procedures for comparison of a set of machine learning models on a single problem have been proposed almost a decade ago.

Different statistical and common-sense techniques have been used to decide whether the differences between the algorithms are real or random. The most frequently used test is the t-test. In this study we choose to select a different approach to validate our model taking into consideration concerns raised against the widely used t-test as usually conceptually inappropriate and statistically unsafe [24].We will use Wilcoxon (1945) signed-ranks test. It is a non-parametric tests which violates the assumption of normally distributed data and is less affected by outliers.

The Wilcoxon signed-ranks test (Wilcoxon, 1945) is a non-parametric alternative to the paired t-test, which ranks the differences in performances of two classifiers for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences.

Let d_i be the difference between the performance scores of the two classifiers on i^{th} out of N data sets. The differences are ranked according to their absolute values; average ranks are assigned in case of ties. Let R^+ be the sum of ranks for the data sets on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2} \sum_{d_i=0} rank(d_i) \tag{2}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2} \sum_{d_i=0} rank(d_i) \tag{3}$$

Let T be the smaller of the sums, $T = min(R^+, R^-)$. Most books on general statistics include a table of exact critical values for T and N up to 25 (or sometimes more).

For a larger number of data sets, the statistics

$$Z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{4}$$

is distributed approximately normally. With $\alpha = 0.05$, the null-hypothesis can be rejected if Z is smaller than -1.96. This means that the statistic Z has fallen into the region of rejection. The p-value translates the statistic Z into a standard normal distance. So, the null hypothesis is rejected if the p-value is less than 0.05 in the standard normal curve.

5.2 Statistical Validation of Models

To compute the statistical significance of the results listed in table 1; we claim the null hypothesis (H_0) to be that there is NO significant difference between

Table 2. Statistical validation results {Z-value(P-value)} for NN3 data set

	GPR	MLP/GPR Ensemble	NCL	NCL/MLP/GPR Ensemble
MLP	-1.1959 (0.2317)	**-3.2143 (0.0013)**	**-2.4745 (0.0133)**	**-4.5216 (0.00001)**
GPR	0(0)	**-2.5750 (0.01)**	**-2.0220 (0.0432)**	**-3.9326 (0.0001)**
MLP/GPR Ensemble		0(0)	-0.1472 (0.8829)	**-2.6612 (0.0078)**
NCL			0(0)	**-3.5447 (0.0004)**

Table 3. Statistical validation results {Z-value(P-value)} for Tourism data set

	GPR	MLP/GPR Ensemble	NCL	NCL/MLP/GPR Ensemble
MLP	-0.8108 (0.4175)	-0.7246 (0.4687)	-1.3387 (0.1807)	**-2.2061 (0.0274)**
GPR	0(0)	**-2.4000 (0.0164)**	-1.5812 (0.1138)	**-2.4862 (0.0129)**
MLP/GPR Ensemble		0(0)	-1.1609 (0.2457)	**-2.4647 (0.0137)**
NCL			0(0)	-0.1374 (0.8907)

the performance of any two possible models used in this work. Tables 2 and 3 show the statistical validation results on the two data sets used in our study.

The highlighted cells indicate that there is a significant difference in performance between the method listed in the row compared to the method listed in the column. For example, for the NN3 data, the results indicate that MLP/GPR, NCL and NCL/MLP/GPR outperform the single models MLP and GPR with a statistical significance. In the same time, although the SMAPE results indicate that the NCL outperforms the MLP/GPR model; there is no statistical significance found for this result. So we can consider that NCL and MLP/GPR are on a tie. On the other hand, a clear conclusion can be drawn that the NCL/MLP/GPR ensemble outperforms both the NCL model and the MLP/GPR model with a significant statistical difference according to Wilcoxon statistical test with 90% significance level. The same result applies to the Tourism data sets, except that in that case NCL vrs. NCL/MLP/GPR were found almost a tie. This leaves us with the general conclusion that the NCL and the combined NCL/MLP/GPR model have proven to be very effective for the application on hand.

6 Conclusions and Future Work

In this work we investigate efficient machine learning techniques and ensemble methods for time series prediction. The application we focus on is forecasting tourist arrivals. We also use the NN3 data in our study because it pocesses similar characteristics like the timeseries obtained from the tourism domain. In this work we devise a combined model of MLP, GPR and NCL to predict time series data. A comparative analysis revealed that the combined model outperforms the single models of the MLP, GPR, NCL and the MLP/GPR model. Currently we are investigating predictor combination under uncertainty. We also intend to investigate other forecasting techniques and models like block/ group training , fuzzy wavelet networks and echo state networks.

Acknowledgment

This work is part of the Data Mining for Improving Tourism Revenue in Egypt research project within the Egyptian Data Mining and Computer Modeling Center of Excellence. We would like to acknowledge the cooperation of the Egyptian Ministry of Tourism MoT. We would also like to thank Dr. Gavin Brown of the University of Birmingham for his advice and help to implement the Negative correlation Algorithm.

References

1. Brockwell, P.J., Davis, R.A.: Introduction to Time Series and Forecasting. Springer, Heidelberg (2002)
2. Verdes, P.F., Granitto, P.M., Navone, H.D., Ceccatto, H.A.: Frost prediction with machine learning techniques. In: The VI^{th} Argentine Congress on Computer Science (2000)
3. Lendasse, A., Francois, D., Wertz, V., Verleysen, M.: Vector quantization: a weighted version for time-series forecasting. Future Gener. Comput. Syst. 21(7), 1056–1067 (2005)
4. Cai, X., Zhang, N., Venayagamoorthy, G.K., Donald, I., Wunsch, C.: Time series prediction with recurrent neural networks trained by a hybrid pso-ea algorithm. Neurocomput. 70(13-15), 2342–2353 (2007)
5. Cellier, F., Nebot, A.: Multi-resolution time-series prediction using fuzzy inductive reasoning. In: Proceedings of the IJCNN 2004: International Joint Conference on Neural Networks, Budapest, Hungria, vol. 2, pp. 1621–1624 (2004)
6. Faming, L.: Bayesian neural networks for nonlinear time series forecasting. Statistics and Computing 15(17), 13–29 (2005)
7. Cheng, H., Tan, P.-N., Gao, J., Scripps, J.: Multistep-ahead time series prediction. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS, vol. 3918, pp. 765–774. Springer, Heidelberg (2006)
8. Ahmed, N.K., Atiya, A., Gayar, N.E., El-Shishiny, H.: An empirical comparison of machine learning models for time series forecasting. Econometric Reviews (2009) (accepted for publication)

9. Chan, P.P., Zeng, X., Tsang, E.C., Yeung, D.S., Lee, J.W.: Neural network ensemble pruning using sensitivity measure in web applications. In: IEEE International Conference on Systems, Man and Cybernetics, 2007. ISIC, pp. 3051–3056 (2007)
10. Brown, G., Yao, X.: On the effectiveness of negative correlation learning. In: Proceedings of First UK Workshop on Computational Intelligence, pp. 57–62 (2001)
11. Wichard, J., Ogorzalek, M.: Time series prediction with ensemble models. In: International Joint Conference on Neural Networks (IJCNN), Budapest, pp. 1625–1629.
12. Kuncheva, L.: Combining pattern classifiers: Methods & Algorithms. Wiley-interscience, Hoboken (2004)
13. Navone, H.D., Verdes, P.F., Granitto, P.M., Ceccatto, H.A.: A learning algorithm for neural networks ensembles. In: ASAI 2000: Proceedings of the Argentine Symposium on Artificial Intelligence, vol. 12, pp. 70–74 (2001)
14. Ahmed, N., Atiya, A., Gayar, N.E., El-Shishiny, H.: A combined neural network/gaussian process regression time series forecasting system for the nn3 competition (2007), http://www.neural-forecasting-competition.com/NN3/results.htm
15. Andrawis, R.R., Atiya, A., El-Shishiny, H.: Forecast combination model using computational intelligence/linear models for the nn5 time series forecasting competition (2008), http://www.neural-forecasting-competition.com/results.htm
16. Liu, Y., Yao, X., Higuchi, T.: Evolutionary ensembles with negative correlation learning. IEEE Transactions on Evolutionary Computation 4, 380–387 (2000)
17. Lin, M., Tang, K., Yao, X.: Selective negative correlation learning algorithm for incremental learning. In: IJCNN, pp. 2525–2530 (2008)
18. Chan, Z., Kasabov, N.: Fast neural network ensemble learning via negative-correlation data correction. IEEE Transactions, Neural Networks 16, 1707–1710 (2005)
19. Eastwood, M., Gabrys, B.: Lambda as a complexity control in negative correlation learning. In: NiSIS 2006 Symposium: 2^{nd} European Symposium on Nature-inspired Smart Information Systems (2006)
20. Armstrong, J.S.: Combining forecasts. In: Principles of Forecasting: A Handbook for Researchers and Practitioners, pp. 417–439. Kluwer Academic Publishers, Dordrecht (2001)
21. Boyle, P., Frean, M.: Dependent gaussian processes. In: Advances in Neural Information Processing Systems, vol. 17, pp. 217–224. MIT Press, Cambridge
22. Rasmussen, C.E.: Gaussian processes for machine learning. MIT Press, Cambridge (2006)
23. Brown, G., Wyatt, J.L., Kaelbling, P.: Managing diversity in regression ensembles. Journal of Machine Learning Research 6 (2005)
24. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)

Diversity-Based Classifier Selection for Adaptive Object Tracking

Ingrid Visentini[1], Josef Kittler[2], and Gian Luca Foresti[1]

[1] Dept of Mathematics and Computer Science, University of Udine, 33100 Udine, Italy
[2] CVSSP, University of Surrey, Guildford, Surrey, UK, GU2 7XH

Abstract. In this work we propose a novel pairwise diversity measure, that recalls the Fisher linear discriminant, to construct a classifier ensemble for tracking a non-rigid object in a complex environment. A subset of constantly updated classifiers is selected exploiting their capability to distinguish the target from the background and, at the same time, promoting independent errors. This reduced ensemble is employed in the target search phase, speeding up the application of the system and maintaining the performance comparable to state of the art algorithms. Experiments have been conducted on a Pan-Tilt-Zoom camera video sequence to demonstrate the effectiveness of the proposed approach coping with pose variations of the target.

1 Introduction

It is well known that ensemble methods' aim is to aggregate multiple learned models to improve the accuracy of classification. Boosting, bagging and other forms of classifiers combination [8,17,22] give an experimental confirmation and a theoretical explanation that diverse hypotheses joined together produce a strong ensemble, whose error is reduced with respect to the average error of the members.

At the same time, the diversity concept arises from the intuition that a set of very dissimilar classifiers would perform better than a single good decision maker, because its error is compensated by the decisions of the others [12]. Intuitively, the more diverse are the classifiers, the wider is the knowledge and the more tolerant is the ensemble to unpredictable events. As equal classifiers will produce the same (redundant) output, the combination of the responses of several classifiers is useful when they disagree on some inputs. We refer to this measure of disagreement, which initially appeared under the name of *ambiguity* in [10], as *diversity*.

The construction of a classifier ensemble is deemed to take advantage of the diversity of its components [4,7,3]; the empirical explanation has been deduced from experiments [13,6], and the use of diversity in designing ensembles has been intensively analysed [19,5,2,20]. However, despite all the work done to date, there is not a concordant definition and formalization of diversity, but only different representations of the same intuition [11,18]. It is acknowledged that the diversity measures can be categorised in two types: pairwise, when a measure considers a couple of classifiers, and non–pairwise, when the diversity refers to the whole ensemble and its performance. Yule's Q statistics [21], the correlation coefficient and the disagreement measure, for instance, belong to the first set, while Kohavi–Wolpert's measure [9] or Kuncheva's entropy [13] represent the second group.

In this work we propose to exploit a criterion reminiscent of the Fisher linear discriminant as a pairwise diversity measure for the construction of an effective selection of online trained classifiers. This criterion is used to select a subset of classifiers, promoting those with independent errors; the resulting ensemble is then employed to track via classification a moving object in a video sequence, following the idea pioneered in [1]. Unlike most of the literature on classifiers, in our case the ensemble does not require any information on the data; it is initially built on-the-fly with random hypotheses, and then updated with significant information.

Starting with a minimal set of training examples, the experts pool is updated with other patterns coming from the tracking phase: the target found at time $(t-1)$ is employed as a positive sample to update the ensemble parameters at time t. The training set is thus collected as two observations at a time, one for the target and one for the negative sample. This mechanism offers the advantage of selecting fresh and constantly updated classifiers at each step, maintaining the knowledge of the ensemble coherent with the object appearance and, at the same time, allowing to select the most appropriate classifiers in that context. Compared with similar methods, like the Online Boosting algorithm [15] that implicitly promotes diversity between classifiers modifying at each step their weight in the linear combination, the proposed technique allows a dynamic replacement of the participating classifiers without affecting the overall performance.

Preliminary experiments conducted on outdoor video sequence demonstrate that this expert fusion framework can be employed as a robust tracking system that copes with pose variations while following moving objects in a dynamic environment. Moreover, our selection strategy saves the computational cost when compared to the Online Boosting approach, keeping the accuracy comparable.

2 Proposed Solution

Given an ensemble $\mathcal{S}$ of R binary classifiers $\{s_1, s_2, \ldots, s_R\}$ and a set of vector-valued samples X, so that $s_r : X \to \{+1, -1\}$, we define the average prediction of the ensemble at time t on a sample $\mathtt{x}$ as the average of the individual scores (mean rule)

$$s_t(\mathtt{x}) = \frac{1}{R}\sum_{r=1}^{R} s_{r,t}(\mathtt{x}) \tag{1}$$

The outputs of this classifier on a training set of samples $(\mathtt{x}_n, y_n)$ with $n = 1, \ldots, N$ can be divided considering the class $y_n \in Y = \{-1, +1\}$ of the sample $\mathtt{x}_n \in X$. At time t, these two separate sets model two distinct probability density functions $P(s_t(\mathtt{x})|y)$ with priors $P(y)$, means $\mu_t^y = \mathbb{E}(s_t)$ and variances $(\sigma_t^y)^2$. For each class $y \in Y$, we can define the variance of the ensemble of (1) as

$$Var^y(s_t) = \mathbb{E}\left\{(s_t(\mathtt{x}) - \mu_t^y)^2\right\} = \mathbb{E}\left\{\left[\frac{1}{R}\sum_{r=1}^{R}(s_{r,t}(\mathtt{x}) - \mu_{r,t}^y)\right]^2\right\}$$

$$= \frac{1}{R^2}\mathbb{E}\left\{\sum_{r=1}^{R}(s_{r,t}(\mathrm{x}) - \mu_{r,t}^{y})^2\right\} + \frac{2}{R^2}\mathbb{E}\left\{\sum_{r=1}^{R}\sum_{j>r}^{R}(s_{r,t}(\mathrm{x}) - \mu_{r,t}^{y})(s_{j,t}(\mathrm{x}) - \mu_{j,t}^{y})\right\}$$
$$= \frac{1}{R^2}\left\{\sum_{r=1}^{R}\mathbb{E}\left[(s_{r,t}(\mathrm{x}) - \mu_{r,t}^{y})^2\right]\right\}$$
$$+ \frac{1}{R^2}2\left\{\sum_{r=1}^{R}\sum_{j>r}^{R}\mathbb{E}\left[(s_{r,t}(\mathrm{x}) - \mu_{r,t}^{y})(s_{j,t}(\mathrm{x}) - \mu_{j,t}^{y})\right]\right\} \tag{2}$$

Analysing (2), we can observe that the first bracketted term is the sum of the variances of the R classifiers

$$\sum_{r=1}^{R}\mathbb{E}\left\{(s_{r,t}(\mathrm{x}) - \mu_{r,t}^{y})^2\right\} = \sum_{r=1}^{R}\mathbb{E}\left\{(s_{r,t}(\mathrm{x}) - \mathbb{E}(s_{r,t}))^2\right\} = \sum_{r=1}^{R}\left(\sigma_{r,t}^{y}\right)^2 \tag{3}$$

The second addend takes the shape of the covariance between the ensemble classifiers

$$\begin{aligned} &2\sum_{r=1}^{R}\sum_{j>r}^{R}\mathbb{E}\left[(s_{r,t}(\mathrm{x}) - \mu_{r,t}^{y})(s_{j,t}(\mathrm{x}) - \mu_{j,t}^{y})\right] \\ &= 2\sum_{r=1}^{R}\sum_{j>r}^{R}\mathbb{E}\left[(s_{r,t}(\mathrm{x}) - \mathbb{E}(s_r))(s_{j,t}(\mathrm{x}) - \mathbb{E}(s_j))\right] \\ &= 2\sum_{r=1}^{R}\sum_{j>i}^{R}\mathrm{Cov}(s_{r,t}(\mathrm{x}), s_{j,t}(\mathrm{x})) \\ &= \sum_{r=1:R}\ \sum_{(j=1:R)\wedge(j\neq r)}\mathrm{Cov}(s_{r,t}(\mathrm{x}), s_{j,t}(\mathrm{x})) \end{aligned} \tag{4}$$

The first and the second part of (2) are thus respectively the diagonal and off-diagonal side of the covariance matrix of the classifier ensemble. They are indicators of the goodness of the classifier with respect to the training set. The first term is greater than zero by definition. The second term is a measure of the relationship between pairs of classifiers, and it can be positive or negative; this last condition is the most desirable. When considering the covariance matrix, the element in position (i, j) is denoted by $\mathbb{E}\left[(s_i^y(\mathrm{x}) - \mathbb{E}(s_i^y(\mathrm{x})))(s_j^y(\mathrm{x}) - \mathbb{E}(s_j^y(\mathrm{x})))\right]$.

To speed up the application of the ensemble and to improve its performance, we wish to reduce its dimensionality and its complexity by choosing a subset $\mathcal{S}_M$ of classifiers $s_1, s_2, \ldots, s_M$ from the initial pool $\mathcal{S}$ so that $M << R$, reducing optimally the number of classifiers according to a certain evaluation criterion (objective function). This selection has two main advantages: 1) the decrement in computational cost during the application; 2) the resulting ensemble is composed of low error classifiers, that improve accuracy of the system.

In order to form a smaller set $\mathcal{S}_M$ with equal or better performance than the entire pool of predictors, we add classifiers to the current set choosing at each step $m = 1, \ldots, M$ the one that maximises the objective function J

$$s_{m,t} = \underset{s \notin \mathcal{S}_{m-1}, s \in \mathcal{S}}{\arg\max} \left[J(\mathcal{S}_{m-1} \cup s) \right] \tag{5}$$

where $\mathcal{S}_0 = \emptyset$. The output of the selected set becomes $\langle s \rangle_t(\mathrm{x}) = \frac{1}{M} \sum_{m=1}^{M} s_{m,t}(\mathrm{x})$.

In our case, the objective function has two facets: on one hand, the minimization of the covariance value between the classifiers of the selected set for both classes in Y guarantees that the classifiers are conditionally independent. In this way, the diversity acts as an effective selection tool. On the other hand, we want to maximize the separability of the class distributions, providing a good ensemble of decision makers. Without loss of generality, we can assume that, for every expert s_r, $(\mu_r^+ > \mu_r^-)$; the class separability can then be represented as the distance between the means $(\mu_r^+ - \mu_r^-)$. We can measure the linear discriminating power of the classifier ensemble $\mathcal{S}$ through the ratio of separability and independence

$$\text{F–ratio}_t(\mathcal{S}) = \frac{(\mu_t^+ - \mu_t^-)^2}{(Var^+(s_t) + Var^-(s_t))} \tag{6}$$

Fusing the selection approach (5), and the criteria (6), we can select incrementally the best M decision makers that satisfy

$$s_{m,t} = \underset{s \notin \mathcal{S}_{m-1}, s \in \mathcal{S}}{\arg\max} \left[\text{F–ratio}_t(\mathcal{S}_{m-1} \cup s) \right] \tag{7}$$

This greedy approach, that at every step m adds the most convenient classifier with respect to the selected ones, is based on the Sequential Feature Selection technique; working in real-time with R classifiers does not allow us to apply an exhaustive search over all the possible $2^R - 1$ subsets as proposed for a similar problem in [16].

Algorithm 1 presents the pseudocode for the proposed solution; among all the possible hypotheses the one that provides the highest F–ratio is added greedily to the ensemble. The limit for including classifiers is given by the need to prevent a decrease of the F–ratio value; the final selection is performed for the next search phase.

The main disadvantage of this solution is that the computation is performed regularly at every frame t, in order to remove classifiers that become obsolete when the training set changes during the on-line learning.

3 Experiments

We employed our approach for object detection and tracking in video surveillance applications. In this section, an experiment performed on real-world video sequences is presented to validate the proposed framework. The hardware employed in all the tests is an AMD Athlon64 $3500+$ with 1GB of RAM.

3.1 Features and Classifiers

We employed three different types of features to describe moving objects: Haar features, Local Binary Patterns (LBP) [14], and colour histograms. All the modules have

Algorithm 1. F–ratio based selection

```
Require: Classifiers pool $\mathcal{S}$
Require: Empty selection set $\mathcal{S}_0 = \emptyset$
  for $t = 1, 2, \ldots, T$ do
    // Update the scores of the hypotheses on the positive sample $\mathtt{x}_t^+$
    // and on the negative sample $\mathtt{x}_t^-$
    for all the $R$ hypotheses do
      Probe $\mathtt{s}_{r,t}(\mathtt{x}_t^+), s_{r,t} \in \mathcal{S}$
      Probe $\mathtt{s}_{r,t}(\mathtt{x}_t^-), s_{r,t} \in \mathcal{S}$
    end for
    // Randomly add the first classifier
    $\mathcal{S}_1 \leftarrow \mathcal{S}_0 \cup s_{random} : \{s_{random} \notin \mathcal{S}_0, s_{random} \in \mathcal{S}\}$
    F–ratio$_{curr} \leftarrow 0$
    F–ratio$_{prev} \leftarrow 0$
    // While the F–ratio grows, add classifiers to the selection
    while F–ratio$_{curr} \geq$ F–ratio$_{prev}$ do
      F–ratio$_{prev} \leftarrow$ F–ratio$_{curr}$
      $\mathcal{S}_m \leftarrow \mathcal{S}_{m-1} \cup \arg\max_{\{s \notin \mathcal{S}_{m-1}, s \in \mathcal{S}\}} \{\text{F–ratio}_t(\mathcal{S}_{m-1} \cup s)\}$ as in (6)
      F–ratio$_{curr} \leftarrow$ F–ratio$(\mathcal{S}_m)$
      $m \leftarrow m + 1$
    end while
    Output: Selection set $\mathcal{S}_M$
  end for
```

been implemented in C++ using fast structures, i.e. integral images and integral histograms, to reduce the computational requirements. The LBP operator calculated on a central pixel (x_c, y_c) and 8 neighbours has the form $LBP(x_c, y_c) = \sum_{i=0}^{7} f(v_i, v_c)2^i$ where i is an index over the neighbours of the central pixel, v_i and v_c are the intensity values of the pixel (x_i, y_i) and the central one respectively, and $f(v_c, v_i) = 1$ if $v_c < v_i$, $f(v_c, v_i) = 0$ otherwise. For what concerns colour histogram features, the region of interest is divided in several random rectangles separately calculated; the Bayesian classifier operates on the Gaussian distribution given by the Bhattacharyya distance for positive and negative samples from the template histogram obtained in the first frames. In this work, $s_{r,t}$ is a Naive Bayes classifier that discriminates between the background and the target. The positive (target) and negative (background) samples are represented each by a normal distribution $\mathcal{N}(\mu_r^y, (\sigma_r^y)^2)$ where $y \in Y$. The score at time t is determined by the highest posterior probability given by the Bayes classifier

$$s_{r,t}(\mathtt{x}) = \arg\max_{y \in Y} P(y|\mathtt{x}, \mu_{r,t}^y, (\sigma_{r,t}^y)^2) \tag{8}$$

The means $\mu_{r,t}^y$ and the variances $(\sigma_{r,t}^y)^2$ of the positive and negative distributions associated to the r-th classifier are refreshed at each frame using the values previously calculated in the covariance analysis process; in this way, the decision makers are maintained updated with the target appearance.

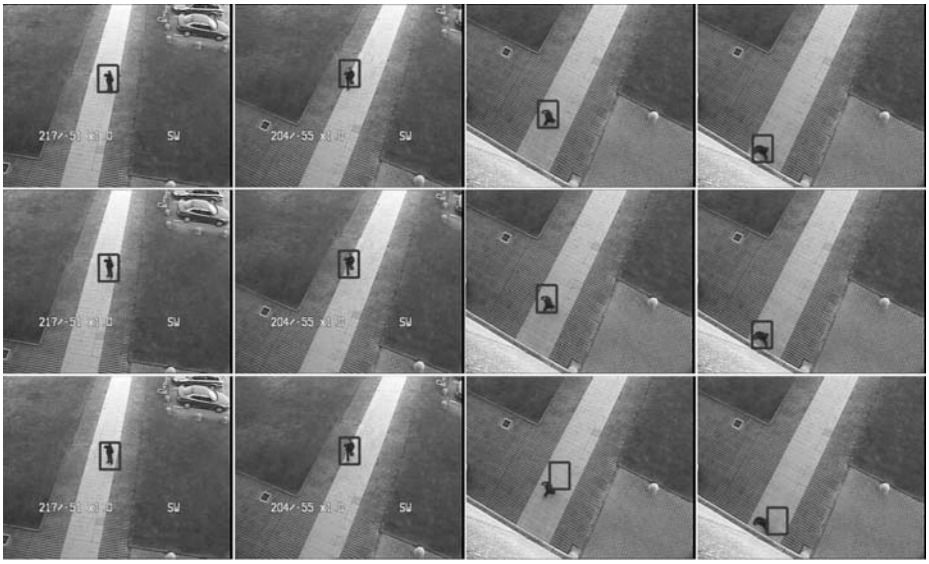


Fig. 1. Comparison of Online Boosting algorithm (top row), proposed approach (second row) and Mean Shift output (third row) on the PTZ sequence at frames 3, 120, 420, 492. The sets are composed of 500 members, but the number of classifiers effectively applied during the search phase in the proposed approach varies as in Fig 2

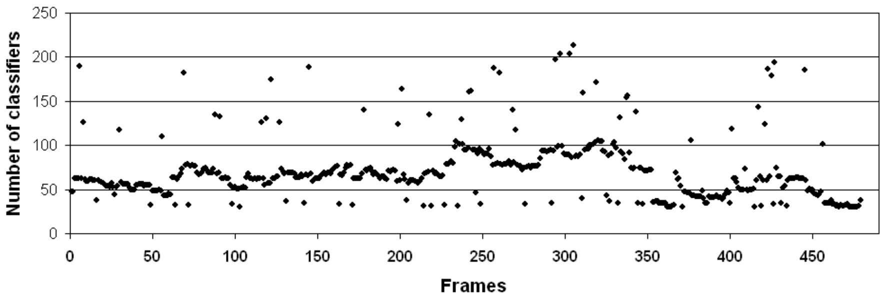


Fig. 2. Number of classifier included in the proposed solution ensemble on the entire PTZ sequence

3.2 Results

We acquired several videos at a resolution of 360×288 pixels with a Pan-Tilt-Zoom (PTZ) camera placed on the top of a building for a total of about 5400 frames. We followed a pedestrian controlling the camera parameters; this configuration was chosen in order to show how detection via classification of a non-rigid object can be performed where a simple background subtraction technique could not be employed.

In figure 1 we present some results taken from the output sequences of three different systems: the Online Boosting algorithm (top row), the proposed approach (second row), and the Mean Shift algorithm (third row). All the systems have been initialized with a target patch obtained by a change detection step. The boosting based ensemble and the proposed one are composed of 500 members each, but in the latter case the number of classifiers effectively involved in the search phase varies, as shown in figure 2. In the last frames of the video sequence, Mean Shift misses the target, while the two frameworks that exploit a classifier set to track the object, enhancing at the same time the diversity between the members, are demonstrated to yield to similar accuracy. This last statement is confirmed also by figure 3, where the ROC curves representing the number of true positives out of false positives, generated by modifying the tolerance threshold while classifying, lead to similar results. The Online Boosting and the F–ratio approach are compared also in terms of distance of the target from the ground-truth (figure 4). Although the localisation error of the proposed tracker is slightly worse, the shifts from the ideal centre of the object are limited to a few pixel in both cases.

Figure 5 shows the advantage of the proposed method. We can observe that the proposed approach has a lower computational cost with respect to the online boosting method and, when employing a few classifiers, than Mean Shift, which is faster for large ensembles. This can be explained because, contrary to the online boosting where all the classifiers are applied at the same time, the dynamic F–ratio driven selection allows to reduce the number of experts employed in the search phase, saving computational time. We can also state that using time consuming features or classifiers in the case of the online boosting framework dramatically increases the total time of computation, while in our case the selection of a limited number of classifiers guarantees that the number of operations is kept low. In our case, the heaviest computational part is represented by the covariance matrix estimation; this $R \times R$ matrix suffers from the curse of dimensionality when the number of classifiers grows. Moreover, the score for a fast update can be explored in future using, for instance, the Cholesky decomposition

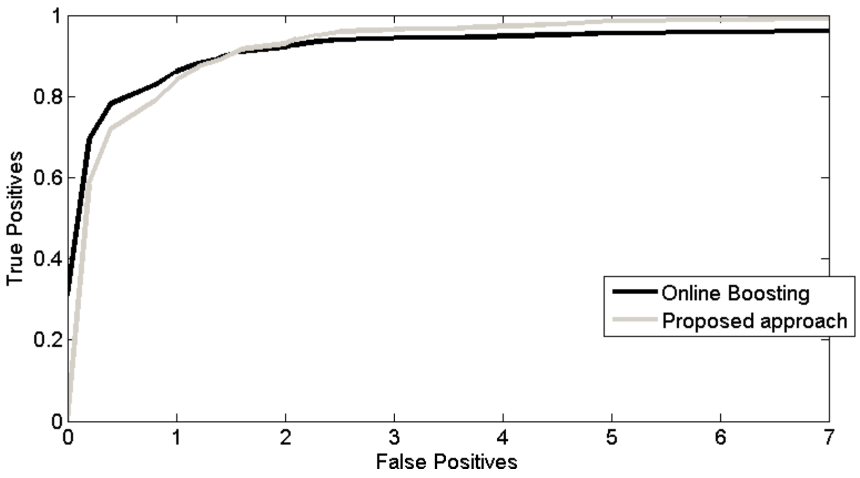
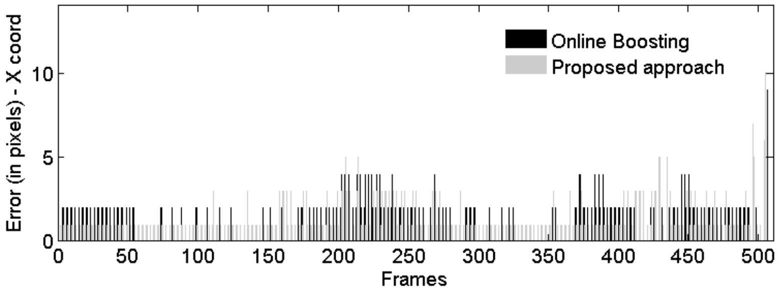
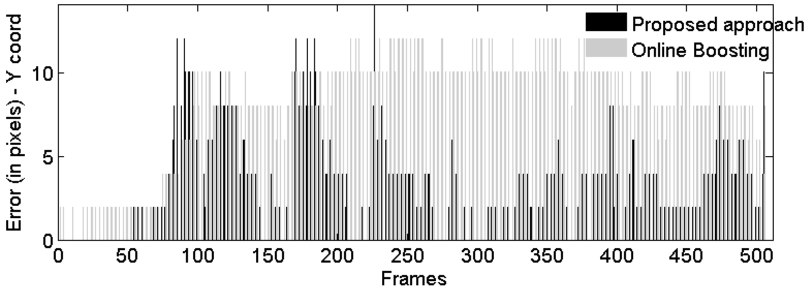


Fig. 3. ROC curves comparison for the Online Bosting and the proposed technique. The ensembles are composed of 500 classifiers and they refer to the sequence of figure 1.



(a)



(b)

Fig. 4. Error in pixels with respect to the ground truth for our approach and for the Online Boosting algorithm [15] on the PTZ sequence: X (a) and Y (b) coordinates.

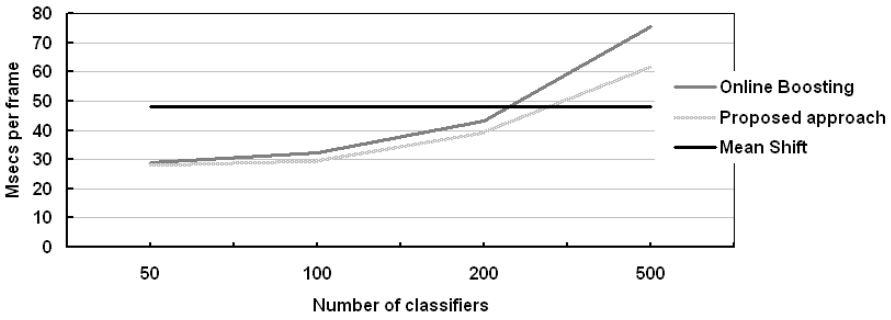


Fig. 5. Running time (in milliseconds) per frame on the sequence of Fig.1 for the three compared methods. Mean Shift obtained the best computational time, followed by the non-optimized proposed approach, and by the online boosting algorithm

instead of calculating the full matrix at every frame. In figure 6 the trend of the means, that represent the expected values of the target pattern (positive samples) and the random background (negative samples) distributions, is drawn.

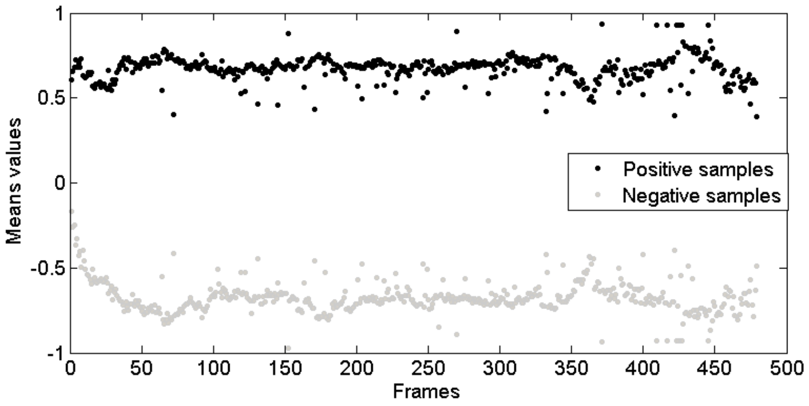


Fig. 6. Means of the positive (target) and negative (background) samples distributions for the proposed approach on the frames of the outdoor video sequence

4 Conclusions

In this work we propose a novel criterion that recalls the Fisher linear discriminant as pairwise diversity measure for the construction of fast and robust classifiers, that are able to distinguish the target and the background classes committing independent errors. This subset is used to track a non-rigid object in a video sequence; to keep updated the knowledge of the ensemble, the classifiers are trained with fresh patterns coming from the tracking phase. Unlike existing methods, such as the Online Boosting algorithm [15] that implicitly promotes the diversity between members by adapting their weights but implies to work always with a fixed number of experts, our approach considers the diversity as an effective criterion to construct a performing ensemble that allows a flexible replacement of classifiers. Moreover, it is suited to operate with computationally expensive features, because the search phase is unburdened by the selection step. Various experiments conducted on a Pan-Tilt-Zoom camera video sequence demonstrate that the described framework is a robust tracking system which copes with pose variation and has performance comparable with the state of the art algorithms.

References

1. Avidan, S.: Ensemble tracking. IEEE Trans. Pattern Anal. Mach. Intell. 29(2), 261–271 (2007)
2. Bian, S., Wang, W.: On diversity and accuracy of homogeneous and heterogeneous ensembles. Int. J. Hybrid Intell. Syst. 4(2), 103–128 (2007)
3. Brown, G., Wyatt, J.L., Harris, R., Yao, X.: Diversity creation methods: A survey and categorisation. Information Fusion 6(1), 5–20 (2005)
4. Chandra, A., Chen, H., Yao, X.: Trade-off between diversity and accuracy in ensemble generation. In: Jin, Y. (ed.) Multi-Objective Machine Learning. Studies in Computational Intelligence, vol. 16, pp. 429–464. Springer, Heidelberg (2006)

5. Golestani, A., Ahmadian, K., Amiri, A., JahedMotlagh, M.-R.: A novel adaptive-boost-based strategy for combining classifiers using diversity concept. In: ACIS-ICIS, pp. 128–134 (2007)
6. Hong, L., Page, S.E.: Diversity and optimality. Research in Economics 98-08-077e, Santa Fe Institute (August 1998)
7. Kapp, M.N., Sabourin, R., Maupin, P.: An empirical study on diversity measures and margin theory for ensembles of classifiers. In: 10th International Conference on Information Fusion, July 2007, pp. 1–8 (2007)
8. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. 20(3), 226–239 (1998)
9. Kohavi, R., Wolpert, D.H.: Bias plus variance decomposition for zero-one loss functions. In: Saitta, L. (ed.) International Conference on Machine Learning, pp. 275–283. Morgan Kaufmann, San Francisco (1996)
10. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. Advances in Neural Information Processing Systems 7, 231–238 (1995)
11. Kuncheva, L.I.: That elusive diversity in classifier ensembles. In: Perales, F.J., Campilho, A.C., Pérez, N., Sanfeliu, A. (eds.) IbPRIA 2003. LNCS, vol. 2652, pp. 1126–1138. Springer, Heidelberg (2003)
12. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles. Machine Learning 51, 181–207 (2003)
13. Kuncheva, L.I., Rodriguez, J.J.: Classifier ensembles with a random linear oracle. IEEE Transactions on Knowledge and Data Engineering 19(4), 500–508 (2007)
14. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7), 971–987 (2002)
15. Oza, N.C.: Online bagging and boosting. In: 2005 IEEE International Conference on Systems, Man and Cybernetics, October 2005, vol. 3, pp. 2340–2345 (2005)
16. Poh, N., Bengio, S.: How do correlation and variance of base classifiers affect fusion in biometric authentication tasks? IEEE Trans. on Sig. Processing 53(11), 4384–4396 (2005)
17. Polikar, R.: Ensemble based systems in decision making. IEEE Circuits and Systems Magazine 6(3), 21–45 (2006) (Third Quarter)
18. Tang, E.K., Suganthan, P.N., Yao, X.: An analysis of diversity measures. Machine Learning 65(1), 247–271 (2006)
19. Wenyao, L., Zhaohui, W., Pan, G.: An entropy-based diversity measure for classifier combining and its application to face classifier ensemble thinning. In: Li, S.Z., Lai, J.-H., Tan, T., Feng, G.-C., Wang, Y. (eds.) SINOBIOMETRICS 2004. LNCS, vol. 3338, pp. 118–124. Springer, Heidelberg (2004)
20. Windeatt, T.: Accuracy/diversity and ensemble mlp classifier design. IEEE Transactions on Neural Networks 17(5), 1194–1211 (2006)
21. Udny Yule, G.: On the association of attributes in statistics. Philosophical Transactions of the Royal Society of London 194, 257–319 (1900)
22. Zhou, Z.-H., Wu, J., Tang, W.: Ensembling neural networks: many could be better than all. Artif. Intell. 137(1-2), 239–263 (2002)

Ensemble Based Data Fusion for Gene Function Prediction

Matteo Re and Giorgio Valentini

DSI, Dipartimento di Scienze dell' Informazione,
Università degli Studi di Milano,
Via Comelico 39, 20135 Milano, Italia
{re,valentini}@dsi.unimi.it

Abstract. The availability of an ever increasing amount of data sources due to recent advances in high throughput biotechnologies opens unprecedented opportunities for genome-wide gene function prediction. Several approaches to integrate heterogeneous sources of biomolecular data have been proposed in literature, but they suffer of drawbacks and limitations that we could in principle overcome by applying multiple classifier systems. In this work we evaluated the performances of three basic ensemble methods to integrate six different sources of high-dimensional biomolecular data. We also studied the performances resulting from the application of a simple greedy classifier selection scheme, and we finally repeated the entire experiment by introducing a feature filtering step. The experimental results show that data fusion realized by means of ensemble-based systems is a valuable research line for gene function prediction.

1 Introduction

The integration of multiple sources of heterogeneous biomolecular data is a key item for the prediction of gene function at genome-wide level. More in general, functional classification of unannotated genes is a central problem in modern functional genomics and bioinformatics [1]. The ever increasing amount of biomolecular data produced in last years as effect of recent advances in high-throughput biotechnologies did not result into a corresponding improvement in gene function prediction accuracy, because the additional complexity introduced by the need to integrate heterogeneous data sources constitutes a serious limiting factor [2]. To deal with this problem, several approaches have been proposed in literature. A first one is based on a direct "vector-space integration" by which different vectorial data are concatenated [3]. Modelling interactions between gene products using graphs and functional linkage networks is another valuable research line, as well as the application of probabilistic graphical models [4]. Kernel methods, by exploiting the closure property of kernels with respect to the sum and other algebraic operators, represent another interesting approach for the integration of biomolecular data [5]. Nevertheless, all these methods suffer of limitations and drawbacks, due to their limited scalability to multiple data sources (i.e. Kernel integration methods based on semidefinite programming [5]), to their limited modularity when new data sources are added (i.e. vector-space integration methods), or when the available biomolecular data are characterized by different structural features (i.e. functional linkage networks and vector-space integration).

A new possible approach is represented by ensemble methods, but not much work has been done to apply classifier integration to gene function prediction [2]. To our knowledge, only few works have been proposed, such as the "late integration" of kernels trained on different sources of data [6], or the Naive-Bayes integration of the outputs of SVMs in the context of the hierarchical classification of genes [7]. Ensemble-based data fusion techniques have been successfully applied in several domains, ranging from biomedical applications [8] to the classification of multisource remote-sensing images [9]. However, there are several reasons to apply ensemble methods in the specific context of genomic data fusion for gene function prediction. At first, biomolecular data differing for their structural characteristics (e.g. sequences, vectors, graphs) can be easily integrated, because with ensemble methods the integration is performed at the decision level, combining the outputs produced by classifiers trained on different datasets. Moreover, as new types of biomolecular data, or updates of data contained in public databases, are made available to the research community, ensembles of learning machines are able to embed new data sources or to update existing ones by training only the base learners devoted to the newly added or updated data, without retraining the entire ensemble. Finally most ensemble methods scale well with the number of the available data sources, and problems related to the addition of newly available sources of biomolecular data can be easily managed.

In this contribution we investigate the effectiveness of different types of ensemble systems in gene function prediction. We also evaluate the effect on the quality of predictions due to the introduction of a simple base classifier selection scheme. We finally repeat the entire experiment introducing a feature selection step. The results are then compared with baseline methods to provide an overview of the potentialities of multiple classifier systems in gene function prediction.

2 Methods

In our experiments, to integrate different sources of biomolecular data, we chose relatively simple methods, such as weighted average combination methods and decision templates. As a second step we considered ensembles based on base learner selection, according to the test-and-select approach, and finally we applied ensembles combined with simple feature filtering methods to reduce the high dimensionality that characterize biomolecular data.

2.1 Ensemble Methods for Biomolecular Data Fusion

Data fusion can be realized by means of an ensemble system composed by learners trained on different "views" of the data and then combining the outputs of the component learners. Each type of data may capture different and complementary characteristics of the objects to be classified and the resulting ensemble may obtain better prediction capabilities through the diversity and the anti-correlation of the base learner responses.

In particular, each type of biomolecular data $B_1, B_2, \ldots, B_T$ is characterized by different features $f_1, f_2, \ldots, f_T$, where T is the number of the available data sources. Thus, an example x is characterized by different sets of features:

$$\mathbf{x} = < \mathbf{x}_{f_1}, \mathbf{x}_{f_2}, \ldots, \mathbf{x}_{f_T} > \tag{1}$$

where $\mathbf{x}_{f_t}$ represents the data relative to the features f_t of a specific data set $B_t \subset X_t$.

A classifier trained on data B_t computes a function $d_{t,j} : X_t \rightarrow [0, 1]$ that estimates the support (e.g. the probability) that a given example x belongs to a specific class ω_j. In our experiments we applied a sigmoid fitting to the output of SVMs, to obtain an estimate of the probability that a given example belongs to a given class [10]. An ensemble combines the outputs of T base learners, each trained on a different type of biomolecular data, using a suitable combining function g to compute the overall support μ_j for a given class ω_j:

$$\mu_j(\mathbf{x}) = g(d_{1,j}(\mathbf{x}_{f_1}), d_{2,j}(\mathbf{x}_{f_2}), \ldots, d_{T,j}(\mathbf{x}_{f_T})) \tag{2}$$

At first, we combine the base classifiers through the classical *weighted average rule*:

$$\mu_j(\mathbf{x}) = \sum_{t=1}^{T} w_t d_{t,j}(\mathbf{x}_{f_t}) \tag{3}$$

In our experiments we computed the weights according to a convex combination rule (w_t^c) and a logarithmic transformation (w_t^{log}):

$$w_t^c = \frac{F_t}{\sum_{t=1}^{T} F_t} \qquad\qquad w_t^{log} \propto log \frac{F_t}{1 - F_t} \tag{4}$$

In both cases we use the F-measure F_t, i.e. the harmonic mean between precision and recall, instead of the classical accuracy, since the gene functional classes are largely unbalanced (positive examples are largely less than negative ones). F_t measures are obtained by "internal" cross-validation on the training data. The ensemble chooses the class ω_j, according to the estimated probability μ_j (eq. 3):

$$D_j(\mathbf{x}) = \begin{cases} 1, & \text{if } \mu_j(\mathbf{x}) > h \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where output 1 corresponds to positive predictions for ω_j and 0 to negatives. A reasonable value for the threshold h is 0.5 (if μ_j estimates probabilities). Note that in this setting an example $\mathbf{x}$ may belong to more than one class (eq. 5), thus modeling the multilabel classification problem that characterizes gene function prediction.

Some base learners trained on specific biomolecular data may incorrectly predict the examples for a given gene functional class for several reasons. For instance certain types of biomolecular data can be informative for some functional classes, but uninformative for others. In order to take into account systematic incorrect answers of certain base learners, *Decision Templates* [11] can represent a valuable approach. In this approach the decision profile DP($\mathbf{x}$) for an instance $\mathbf{x}$ is a matrix composed by the $d_{t,j} \in$[0,1] elements representing the support given by the t^{th} classifier to class ω_j. Decision templates DT_j are the averaged decision profiles obtained from $\mathbf{X}_j$, the set of training instances belonging to the class ω_j:

$$DT_j = \frac{1}{|\mathbf{X}_j|} \sum_{\mathbf{x} \in \mathbf{X}_j} DP(\mathbf{x}) \tag{6}$$

The similarity $\mathcal{S}$ between the decision template DT_j for a class ω_j, $1 \leq j \leq C$, and the decision profile for a given test instance $\mathbf{x}$ is:

$$\mathcal{S}_j(\mathbf{x}) = 1 - \frac{1}{T \times C} \sum_{t=1}^{T} \sum_{k=1}^{C} [DT_j(t,k) - d_{t,k}(\mathbf{x})]^2 \tag{7}$$

and the final decision of the ensemble is computed by assigning the test instance to the class with the largest similarity:

$$D(\mathbf{x}) = \arg\max_j \mathcal{S}_j(\mathbf{x}) \tag{8}$$

For gene prediction we consider two-classes problems, because a gene may belong or not to a given functional class. To simplify the notation, we denote the positive class by 1 and the negative by 2. In this context, exploiting the fact that $d_{t,2}(\mathbf{x}) = 1 - d_{t,1}(\mathbf{x})$, the similarity $\mathcal{S}$ (eq. 7) for the positive and the negative class class becomes:

$$\mathcal{S}_1(\mathbf{x}) = 1 - \frac{1}{T} \sum_{t=1}^{T} [DT_1(t,1) - d_{t,1}(\mathbf{x})]^2 \tag{9}$$

$$\mathcal{S}_2(\mathbf{x}) = 1 - \frac{1}{T} \sum_{t=1}^{T} [DT_2(t,1) - d_{t,1}(\mathbf{x})]^2 \tag{10}$$

and the final decision of the ensemble is:

$$D(\mathbf{x}) = \arg\max(\mathcal{S}_1(\mathbf{x}), \mathcal{S}_2(\mathbf{x})) \tag{11}$$

2.2 Feature Filtering

Feature selection methods can select the most significant features and can reduce the high dimensionality that characterize most biomolecular data.

To reduce the computational complexity we introduce a simple filtering method based on the t-test statistic: More precisely, we applied the two-sample

Welch t-test to verify the null hypothesis $\mathcal{H}_j$ of no difference between the means of feature values of the two given positive and negative sets of genes at a given significance level α. Since the number of features for each data set is in the order of thousands, we need to restate the problem in a multiple hypothesis test setting. In particular we applied the Benjamini and Hochberg (BH) [12] procedure to control the false discovery rate FDR (that is the expected proportion of false positives among the rejected hypotheses). This procedure is applied separately for each data set.

2.3 Base Learner Selection

According to the *test and select* methodology [13], we apply a variant of the "choose the best" technique [14] to select a subset of "optimal" classifiers. More precisely we select the "best" subset of base classifiers (each one trained on a different source of biomolecular data) according to the F-measure estimated by internal cross-validation on the the training set. A high level scheme of the adopted "test and select" procedure is reported below:

1. Separately for each available data, select the most significant features using the two-sample t-test with Benjamini and Hochberg p-value correction (Sect. 2.2).
2. Train the base learners on the heterogeneous data sets filtered according to step 1.
3. Select the n learners with the best F-measure estimated by internal cross-validation on the training set
4. Evaluate the ensembles with the n best learners on a separated test set.

We applied the "test and select" procedure with and without the first step (feature filtering with "corrected" t-test). Note that at step 2 and 3 a base learner model selection can also be performed using cross-validation on the training data. Weighted average rule and decision templates (Sect. 2.1) are the aggregation strategies adopted to combine the output of the base learners.

3 Experimental Setup

We collected several sources of biomolecular data to classify genes of the yeast, an eukaryotic unicellular model organism. In particular we used protein-protein interaction data collected from BioGrid [15] and STRING [16], a collection of physical and genetic interactions obtained from different types of biological experiments and from literature. Moreover we included data to register the presence/absence of a particular protein domain in the proteins encoded by genes comprised in the dataset [17] and the E-value assigned to each gene product to a collection of profile-HMMs computed through the HMMR software toolkit (`http://hmmer.janelia.org`). We considered also homology relationships data using pairwise Smith-Waterman $\log E$ values between all pairs of yeast sequences. Finally we included into our experiment a dataset obtained by the

Table 1. Datasets

Code	Dataset	examples	features	description
D1	Protein domain binary	3529	4950	protein domains obtained from *Pfam* database [17]
D2	Protein domain log-E	3529	5724	Pfam protein domains with log E-values computed by the *HMMER* software toolkit
D3	Gene expression	4532	250	merged data of Spellman and Gasch experiments [18] [19]
D4	PPI - BioGRID	4531	5367	protein-protein interaction data from the *BioGRID* database [15]
D5	PPI - STRING	2338	2559	protein-protein interaction data from [16]
D6	Pairwise similarity	3527	6349	Smith and Waterman log-E values between all pairs of yeast sequences

integration of microarray hybridization experiments published in [18] [19]. The main characteristics of the data sets used in the experiments are summarized in Tab. 1. The genes represented in the datasets under investigation have been associated to functional classes using the functional annotations collected in the Functional Catalogue (FunCat) database version (2.1) [20].

In our experiments we considered only the first level of the hierarchy of FunCat classes, that is the most general and wide 15 functional classes of the overall taxonomy.

We considered the intersection between all the datasets, resulting into a final collection of 1910 yeast genes. In other words we used in our experiments only the genes for which experimental measures were available for all the types of data. Each resulting dataset was randomly split into a training set and a test set (composed, respectively, by the 70% and 30% of the available samples). We performed a 3-fold stratified cross-validation on the training data for model selection, using gaussian SVMs as base learners. We chose the F-measure for both model selection and to evaluate the performances on the separated test set, because most FunCat classes are unbalanced, with positive examples largely lower than negatives.

We then applied a test and select procedure, by choosing the best 2, 3 or 4 classifiers according to the F-measure evaluated by cross-validation on the training set (Sect.2.3). The test and select procedure has been applied with and without feature selection according to a two-sample t-test and a Benjamini and Hochberg correction at 0.05 significance level (Sect.2.2).

4 Results

Tab. 2 summarizes the averages across the performed 15 dichotomic learning tasks of the F-measure, recall, precision and specificity computed on the test sets using respectively:

1. The ensemble methods described in Sect. 2.1 using all the available data sets and base learners
2. The test-and-select procedure outlined in Sect. 2.3.
3. The feature filtering step added before the test-and-select procedure (Sect. 2.2).

Table 2. Summary of ensemble results. L_{best} refers to the best single learner, L_{avg} to the average results of single SVMs; E_{lin} and E_{log} to weighted average combination with respectively linear and logarithmic weights; E_{DT} stands for decision templates ensembles.

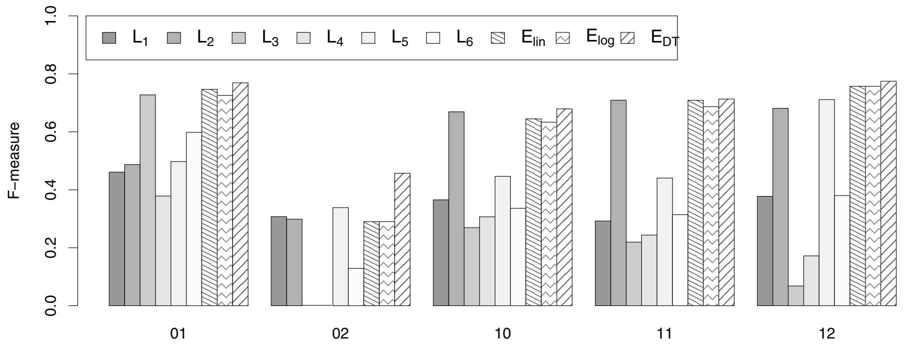
A)	**Results using all the available base learners**				
Metric	L_{best}	L_{avg}	E_{lin}	E_{log}	E_{DT}
F	0.4816	0.3470	0.4403	0.4112	0.5302
rec	0.3970	0.2859	0.3304	0.2974	0.4446
prec	0.6785	0.5823	0.8179	0.8443	0.7034
spec	0.9516	0.9533	0.9798	0.9850	0.9594

B)	**Results with test and select procedures**				
Metric	L_{best}	L_{avg}	E_{lin}	E_{log}	E_{DT}
F	0.4816	0.3470	0.5436	0.5441	0.5698
rec	0.3970	0.2859	0.4793	0.4778	0.5164
prec	0.6785	0.5823	0.6723	0.6591	0.6435
spec	0.9516	0.9533	0.9538	0.9573	0.9447

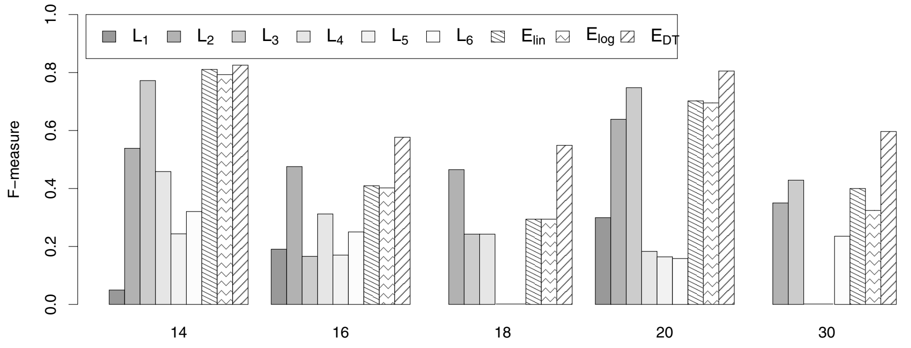
C)	**Results with test and select and feature filtering**				
Metric	L_{best}	L_{avg}	E_{lin}	E_{log}	E_{DT}
F	0.4893	0.2638	0.5175	0.4912	0.6310
rec	0.3841	0.1927	0.3987	0.3711	0.5667
prec	0.7278	0.6141	0.8708	0.9042	0.7439
spec	0.9639	0.9775	0.9841	0.9871	0.9552

L_{best} refers to the best single learner (trained on the D2 protein domain data set, Tab. 1), and L_{avg} to the average results of the single SVMs across all the 6 data sets.

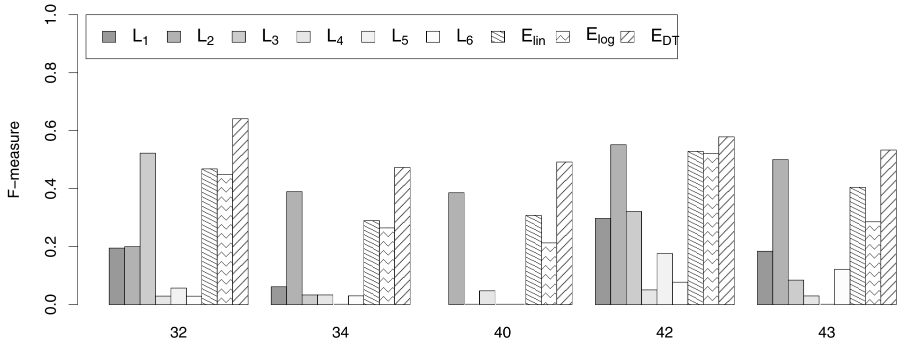
As reported in Tab. 2 A), the performances averaged across all the performed learning tasks are increased by the basic ensemble-based data fusion approaches involving the combination of all the component classifiers. The investigated combination strategies are able, on the average, to outperform the single learners. In particular the Decision Template combiner outperforms the single best classifier in the evaluation of the test set. The simple greedy strategy to test and select the "best" base learners for each classification task significantly enhances the performances of weighted average combination methods (from 0.41 to 0.54 with E_{log}), but also Decision templates gain from this approach (Tab. 2 B). By adding a simple feature selection step to the test and select methods we can observe that only Decision templates are able to improve their performances (Tab. 2 C). Indeed on the average the performances of single learners largely decrease (the F-measure falls from from 0.34 to 0.26), as well as the performances of weighted average ensembles, even if the relative decrement of the latter is lower. In all cases, independently of the adopted ensemble method and with or without feature selection, the ensembles of learning machines largely



(a)



(b)



(c)

Fig. 1. Per class F-measure results of ensemble methods with base learner selection and feature filtering. For each FunCat class, the first six shaded gray bars refer to single learners with feature filtering (from L_1 to L_6); the last three bars (filled with patterns) correspond respectively to weighted average combination with linear (E_{lin}) and logarithmic (E_{log}) weights and decision template (E_{DT}) ensembles. a) Funcat classes 01, 02, 10, 11, 12; b) 14, 16, 18, 20, 30; c) 32, 34, 40, 42, 43.

outperform the average results of the single SVMs. Moreover in most cases ensemble methods outperform also the best single SVM, and in particular decision templates obtain better results than the best single SVM on all the gene function prediction tasks (Fig. 1). It is worth noting that ensemble methods achieve a very high precision (Tab. 2): this is of paramount importance to drive the biological validation of novel predicted genes whose function is unknown or only partially known, in order to reduce the costs of possible false positives.

Each type of biomolecular data set captures different characteristics of genes, and can be informative for some classes but uninformative for the prediction of other classes of genes. From this standpoint we can understand the reasons why simple decision fusion techniques may improve gene function prediction. In particular decision templates seem to better exploit the different characteristics of the available source of biomolecular data. Indeed, through the decision templates, also relatively uncertain or wrong responses of base learners can provide useful information for the decision of the ensemble, especially if this behaviour is consistently maintained across the data. This is confirmed also by the fact that test and select methods with decision templates to combine the output of the selected base learners require on the average more learners than weighted average ensembles (data not shown). Decision templates are thus able to exploit also the characteristics of the less informative base learners to improve the predictions of the overall ensemble.

5 Conclusions

In this work we investigated the effectiveness of ensemble-based data fusion methods on the functional classification of yeast genes. The ensembles are able to outperform the averaged performances of single SVMs in all the gene function prediction tasks, achieving the best results in terms of precision and recall. The performances are further improved by a simple ”choose the best” selection strategy, and a feature filtering method is able to enhance the results of decision templates. Considering the F-measure that summarizes both precision and recall, the experimental results show that data fusion realized by means of ensemble systems is a valuable research line in gene function prediction and that Decision Templates may represent a good choice for biomolecular data integration.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments and suggestions, and gratefully acknowledge partial support by the PASCAL2 Network of Excellence under EC grant no. 216886. This publication only reflects the authors' views.

References

[1] Pena-Castillo, L., et al.: A critical assessment of Mus musculus gene function prediction using integrated genomic evidence. Genome Biology 9 (2008)

[2] Noble, W., Ben-Hur, A.: Integating information for protein function prediction. In: Bioinformatics - From Genomes to Therapies, pp. 1297–1314. Wiley, Chichester (2007)

[3] des Jardins, M., et al.: Prediction of enzyme classification from protein sequence without the use of sequence similarity. In: Proc. of the 5th ISMB, pp. 92–99 (1997)
[4] Karaoz, U., et al.: Whole-genome annotation by using evidence integration in functional-linkage networks. Proc. Natl. Acad. Sci. USA 101, 2888–2893 (2004)
[5] Lanckriet, G., De Bie, T., Cristianini, N., Jordan, M., Noble, W.: A statistical framework for genomic data fusion. Bioinformatics 20, 2626–2635 (2004)
[6] Pavlidis, P., Weston, J., Cai, J., Noble, W.: Learning gene functional classification from multiple data. J. Comput. Biol. 9, 401–411 (2002)
[7] Guan, Y., et al.: Predicting gene function in a hierarchical context with an ensemble of classifiers. Genome Biology 9 (2008)
[8] Polikar, R., et al.: An ensemble based data fusion approach for early diagnosis of Alzheimer disease. Information Fusion 9, 83–95 (2008)
[9] Benediktsson, J., Chanussot, J., Fauvel, M.: Multiple classifier systems in remote sensing: From basics to recent developments. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 501–512. Springer, Heidelberg (2007)
[10] Lin, H., Lin, C., Weng, R.: A note on Platt's probabilistic outputs for support vector machines. Machine Learning 68, 267–276 (2007)
[11] Kuncheva, L., Bezdek, J., Duin, R.: Decision templates for multiple classifier fusion: an experimental comparison. Pattern Recognition 34, 299–314 (2001)
[12] Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. R. Statist. Soc. B 57, 289–300 (1995)
[13] Roli, F., Giacinto, G., Vernazza, G.: Methods for Designing Multiple Classifier Systems. In: Kittler, J., Roli, F. (eds.) MCS 2001. LNCS, vol. 2096, pp. 78–87. Springer, Heidelberg (2001)
[14] Partridge, D., Yates, W.: Engineering multiversion neural-net systems. Neural Computation 8, 869–893 (1996)
[15] Stark, C., et al.: BioGRID: a general repository for interaction datasets. Nucl. Acids Res. 34, D535–D539 (2006)
[16] von Mering, C., et al.: STRING: a database of predicted functional associations between proteins. Nucl. Acids Res. 31, 258–261 (2003)
[17] Finn, R., et al.: The Pfam protein families database. Nucl. Acids Res. 36, 281–288 (2008)
[18] Gasch, P., et al.: Genomic expression programs in the response of yeast cells to environmental changes. Mol. Biol. Cell 11, 4241–4257 (2000)
[19] Spellman, P., et al.: Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomices cerevisiae by microarray hybridization. Mol. Biol. Cell 9, 3273–3297 (1998)
[20] Ruepp, A., et al.: The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucl. Acids Res. 32, 5539–5545 (2004)

A Cascade Multiple Classifier System for Document Categorization

Jian-Wu Xu[1], Vartika Singh[2], Venu Govindaraju[2], and Depankar Neogi[1]

[1] Copanion Inc., Andover, MA 01810, USA
{jxu,dneogi}@copanion.com
http://www.copanion.com

[2] Center for Unified Biometrics and Sensors, University at Buffalo, USA
{vartika,venu}@cubs.buffalo.edu

Abstract. A novel cascade multiple classifier system (MCS) for document image classification is presented in the paper. It consists of two different classifiers with different feature sets. The proceeding classifier uses image features, learns physical representation of the document, and outputs a set of candidate class labels for the second classifier. The succeeding classifier is a hierarchical classification model based on textual features. The candidate labels set from the first classifier provides subtrees for the second classifier to search in the hierarchical tree and derive a final classification decision. Hence, it reduces the computational complexity and improves classification accuracy for the second classifier. We test the proposed cascade MCS on a large scale set of tax document classification. The experimental results show improvement of classification performance over individual classifiers.

Keywords: Document Classification, Multiple-classifiers, Classifier Combination.

1 Introduction

With the exponential proliferation of documents, document image classification becomes an increasingly important step in office automation, digital libraries and many other applications. It provides automatic indexing, archiving and retrieving documents and facilitates higher-level document analysis. Many machine learning algorithms have been used in document image classification such as support vector machine (SVM), K-nearest-neighbor (KNN), latent conditional independence model, decision tree, neural networks and others [1]. These classifiers operate on one of image features, textual features from OCR (optical character recognition) and document layout structure features or some combination depending on the specific problems. Most of document classification work focuses on developing state-of-the-art single classifier.

Recently there has been increasing interest in combining multiple classifiers in order to improve the performance of a given document image classification system. Héroux *et al.* present three classifiers for form categorization using KNN and multi-layer neural networks with image features and tree matching classifier with physical layout features [2]. A voting method of combining two competing classifiers is proposed in [3] to classify business letters based on content features.

There are mainly three different types of architectures in multiple classifier systems, namely, cascade [4], parallel [5] [6] and mixture of both. Cascade multiple classifier systems are a certain form of multi-stage classification where the succeeding classifier takes the output from the proceeding one as input in a cascading fashion. Parallel multiple classifier systems combine individual independent classifiers to obtain final output. Most of multiple classifier system work has been focused on parallel structure partially because it is much easier to train each individual classifiers with its own features independently and fuse the outputs than to modify classifiers in order to fit into cascading scenario. Many classifier combination rules have been proposed in the literature such as sum, product, median, majority vote, Bayesian, Dempster-Shafer and other rules [5] [6] [7]. Rank order statistics rules (e.g. min/max) find better performance against outliers than the sum rule [7]. Behavior-knowledge space method uses prior statistical knowledge of individual classifiers to derive the best final decision [8]. While behavior-knowledge space method relies on the global behavior of individual classifiers, a combination approach using local accuracy of each classifier in small regions of feature space surrounding an unknown test sample is presented in [9].

In information retrieval which is related to document image classification research, several researchers have achieved improvements in classification accuracy via the combination of different classifiers for text categorization. Larkey and Croft use weighted linear combinations of classifier ranks [10]. Hull *et al.* propose linear combinations of probabilities or log odds scores [11]. A linear combination of normalized scores is presented in [12]. Bennett *et al.* introduce a probabilistic method for combining classifiers that considers the context sensitive reliabilities of contributing classifiers [13]. Most of combination methods in text classification research field are parallel approaches and use same type of textual features for all classifiers.

In document classification, image features offers a good characterization of physical representation of the document image. Image features are also fast to implement since it does not need document layout analysis as required in OCR text extraction. But image features alone do not capture the whole characteristics of document because it is the semantics that decides the class for the document. On the other hand, OCR textual features might be noisy due to poor image quality. The challenge is how to combine these two types of features together to improve classification performance. In this paper, we present a cascade multiple classifier system for document image classification. The proposed MCS is composed of two classifiers where the first one uses image features and the succeeding classifier takes the proceeding classifier output and textual features to derive a final classification output. Classifier with image features will produce a set of candidate class labels. This set will enable the second classifier to concentrate a small search space to derive a final output. This is realized in the second classifier with a hierarchical classification model and the hierarchical tree can be easily divided into subtrees according to the candidate class labels set from the proceeding classifier.

The rest of paper is organized as follows. We will describe the proposed cascade multiple classifier system and the individual classifiers with image and textual features respectively in Sec. 2. In Sec. 3, we apply the proposed method in a large scale tax document classification problem. We conclude our work in Sec. 4.

2 Cascade Multiple Classifier System

In this section, we will explain the proposed cascade multiple classifier system and the individual classifiers. The overall architecture is given in Fig. 1 where Classifier 1 and Classifier 2 are connected in cascade approach with image and textual features inputs respectively. Classifier 1 uses image features of the document and Classifier 2 employs textual features from OCR texts. For a single-label multi-class classification problem, a given test document can be one of any L class labels $\{Y_1, Y_2, \ldots, Y_L\}$ at the input to the multiple classifier system. After it passes through Classifier 1, the candidate label set reduces to $\{Y_i, \ldots, Y_m\}$ where the cardinal number is less than L. Given this reduced candidate label set and textual features, Classifier 2 reaches the final class label $\{Y_j\}$ for the test document.

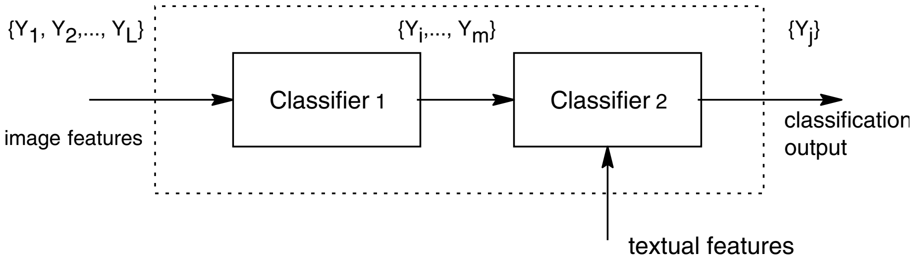


Fig. 1. Cascade multiple classifier system

2.1 Classifier Based on Image Features

Classification using image features has long been an active research topic in document image classification [1]. Image features are either extracted directly from the whole page or from a segmented region. Recently, Sarkar proposed to use 5 dimensional thresholded Viola-Jonese rectangular features with a latent conditional independence model to classify tax documents [14]. Shin *et al.* presented a decision tree classifier based on image features such as percentages of text and non-text content regions, column structures to classify documents [15].

Classifier 1 in our cascade multiple classifier system in Fig. 1 uses local image features, represents features in vector space, and applies KNN classifier to classify document. Fig. 2 shows the basic idea of Classifier 1.

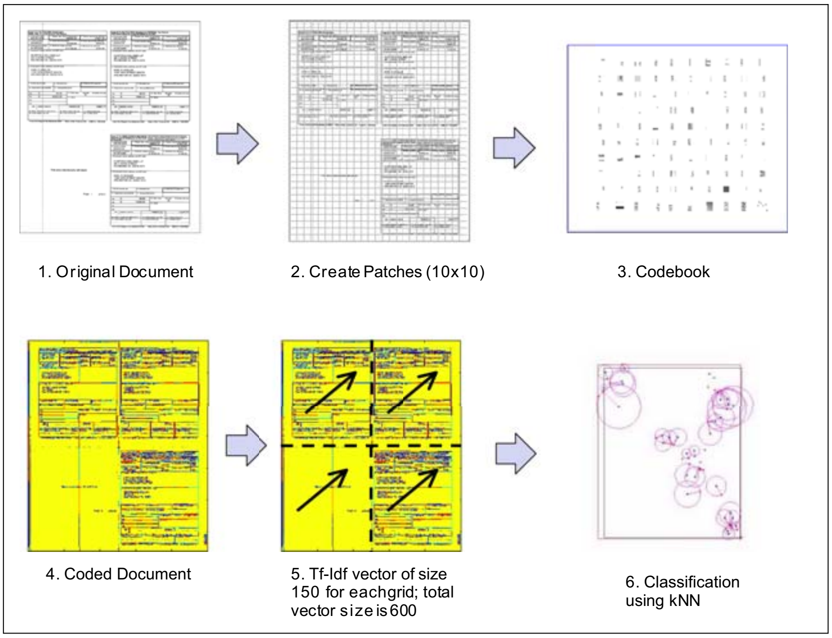

1. Original Document
2. Create Patches (10x10)
3. Codebook
4. Coded Document
5. Tf-Idf vector of size 150 for eachgrid; total vector size is 600
6. Classification using kNN

Fig. 2. Classifier 1 with image features

A document is represented as a typical bag-of-words model in the image domain. The text documents when seen as an image do not have much variation in terms of colors or texture. To make the task simpler, we decide to deal with only binarized image. First, a code book is generated using 7 randomly selected documents. Each of these documents is divided into 10-by-10 pixel blocks (we call these chunks patch). K-means algorithm is applied on all these blocks to generate 150 clusters. The mean of these clusters is taken as the representative codeword for that cluster (shown in step 3 in Fig. 2). The number 150 is reached after observing the results from performing the k-means on blocks from 120 to 170 clusters. Blocks in the original image are replaced by closest blocks in the code word dictionary. If the reconstructed image (step 4 in Fig. 2) is not a very good visual representation of the original, then that experimental cluster number is increased or decreased by 5. 150 codes obtained from the sample 7 documents are enough to represent the entire population of binarized text documents.

Each encoded document is divided into four quadrants (step 5 in Fig. 2). As we are representing the document image following the traditional vector-space model, those 150 patches will form our vocabulary numbered from 1 to 150. A vector is formed for each quadrant following the inverse TF-IDF (term frequency-inverse document frequency) model. Our experiment is limited to text documents where most of the dissimilarity is seen not in the major central area, but mainly around the large area of four outer corners. Four vectors are sufficient to takes

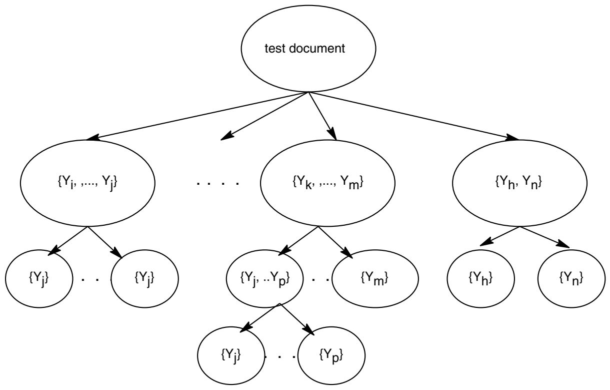


Fig. 3. Hierarchical structure of Classifier 2

this dissimilarity into account. These four vectors are concatenated end-to-end to form the feature vector for the document image.

A set of labeled vectors are collected to form our training set. We obtain clusters representing image of similar document structure with spectral clustering algorithm. We kept increasing the number of clusters so as to achieve minimum variance in any given cluster. We decided upon the variance size by computing the variance of each cluster, and then going upwards in each cluster till the document images started differing too much from their neighbors in the cluster. Since documents from different labeled classes may look similar, a cluster may have more than one labels. A test document is converted to the feature vector form and its Euclidean distance is computed from each of the clusters. The label(s) of the closest cluster(s) are assigned to the document. We can either choose a fix number of labels as our candidate set or a varying number till a threshold is met. We will empirically choose one method in the experiment followed.

2.2 Hierarchical Classifier Based on Textual Features

The second classifier in Fig. 1 is a hierarchical classification model consisting of multi-class and binary classifiers based on textual features [16]. The multi-layer hierarchical structure of classification model derives from the intrinsic hierarchies of given documents. The textual features are vector representation of TF-IDF of OCR texts after χ^2-statistics feature selection.

The hierarchical classification model is illustrated in Fig. 3 where different classes are grouped together in different levels of tree according to textual content of the document image class. The leaves are the final single labels. We apply binary Regularized Least Square (RLS) classifiers to the middle level of hierarchical tree and KNN classifiers to the bottom classes. In each level of tree above the bottom, we classify the test document by multiple *one-against-rest* binary RLS classifiers. Each RLS classifier is constructed for one of the several categories. The l-th RLS classifier is trained on the whole training data set to classify the members of class l against the rest. Each level classification produces a single candidate category for the next level processing. Based on the output from the upper level, another set of RLS classifiers are invoked to further classify the document into smaller set of categories. When the process reaches the bottom of the hierarchical tree, the corresponding category KNN classifier is invoked to classify the document into the final required class. In the case there are multiple candidate categories from the middle levels, we pick the one with highest confidence value.

Classifier 2 with textual features is suitable to incorporate the output information from Classifier 1 in Fig. 1 because it is organized in a hierarchical model. Based on the Classifier 1 output, Classifier 2 can be adapted to only focus on those categories from the candidate label outputs and prune quickly in the hierarchical tree.

2.3 Combination Method

Classifier 1 in the cascade MCS reduces the candidate labels set as shown in Fig. 1. For a given test document, Classifier 1 can either output a fix number of candidate labels or a varying number for different documents. In both cases, Classifier 2 will only focus on those candidates. Specifically, in the hierarchical tree, Classifier 2 will only invoke RLS classifiers for those categories containing candidates from Classifier 1. This approach speeds up the computation in Classifier 2. In some scenarios, only KNN classifier will be used without any RLS classifiers if the candidate labels are all inside one category in the middle level of the tree. KNN classifiers produce a rank-ordered labels and the first one appearing inside the candidate label set from Classifier 1 is chosen as the final output. By incorporating the output information from Classifier 1, the classification accuracy will also increase since the candidate label set reduces the searching space in Classifier 2. However if the true label in case is not in the output candidates set produced by Classifier 1, Classifier 2 will certainly make an error. Therefore, a balanced number of candidates set is important as trade-off between computational complexity and classification accuracy. We will use cross-validation method to choose an optimal number of candidate labels set for Classifier 1 according to the classification accuracy as demonstrated in the experiments.

3 Experimental Results

In this section, we apply the proposed cascade multiple classifier system to individual income tax documents categorization problem where class labeling is based on texts of document. we have collected the most comprehensive tax documents so far in the literature. A large-scale individual income tax related documents are collected through varies accounting firms across US for the 2008 tax season. All these pages have been manually labeled by professionals. The individual income tax documents are organized in a hierarchical structure, e.g. documents issued by US IRS (Internal Revenue Services), organizer (forms containing

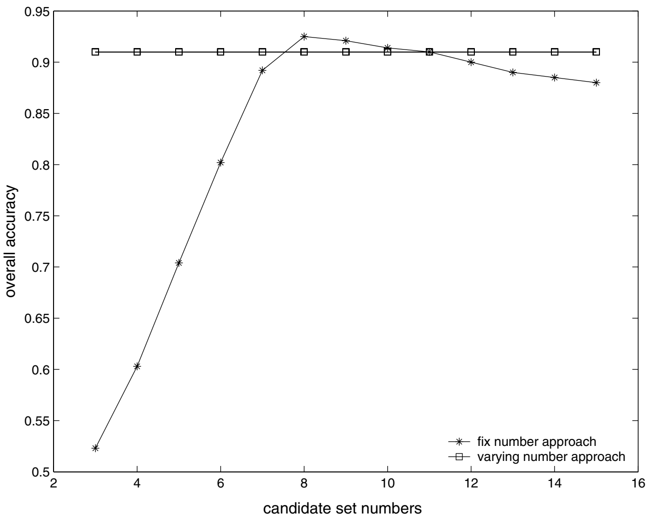


Fig. 4. Overall accuracy versus candidate set sizes

client tax and personal information) and others can be formed into multiple categories and each category can be further divided into small sets. This makes the hierarchical classification model suitable for the task [16]. On the other hand, different class forms have different physical image features which can improve the performance of Classifier 2. It is a single-label multi-class problem with 27 different classes in total.

We first clean up noise and de-skew for bad-quality pages. Then we apply OCR to obtain texts and image processing to extract image features explained in Sec. 2.1 from documents. Most of the numerals are deleted since they do not provide useful information for classification purpose except for those associated with class labels. We also remove stop words and extreme low frequency terms. After all these pre-processings, we randomly select 70% of the overall documents for training, 10% for cross-validation purpose, and remaining 20% for testing.

Classifier 1 and Classifier 2 in the cascade MCS can be trained independently since they are based on different types of features. As explained in Sec. 2.1, Classifier 1 can provide either a fix number of candidate labels for all test documents or different numbers depending on the image feature of the given document. Therefore it is necessary to choose an optimal strategy for the candidate labels set. In the first experiment, we use 10% cross-validation data set to empirically select the best approach. We use the overall accuracy, defined as the percentage of correctly classified data, to compare varying number approach and fix number approach with different candidate set sizes. Fig. 4 plots the overall accuracy versus candidate set sizes. For convenience purpose, we plot the overall accuracy for the varying number method and the one for the fix number method of different candidate set sizes in the same figure. As shown in the figure, candidate set with 8 labels offers the best overall accuracy. As the set size decreases,

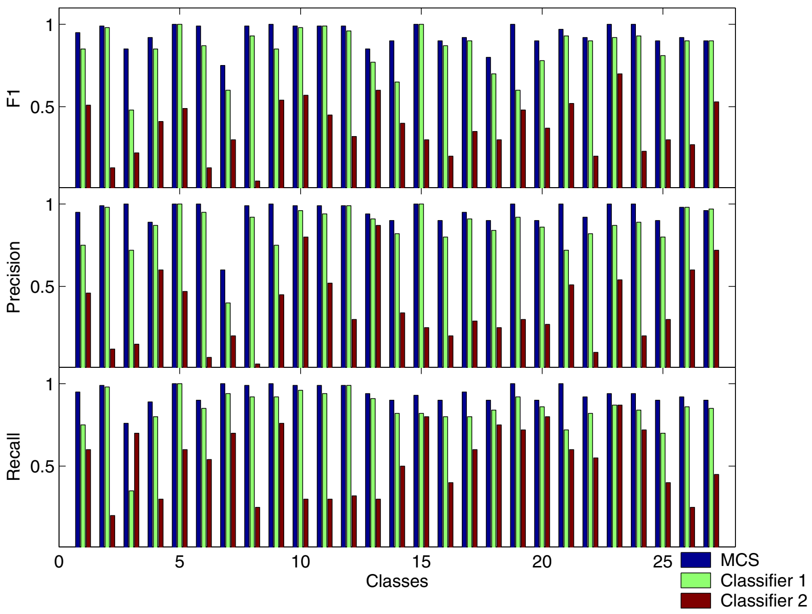


Fig. 5. Performance comparison

the performance drops sharply. While as the size increases, it approaches to the overall accuracy of the Classifier 2 alone. Therefore, we will choose to output 8 candidate labels from Classifier 1 for the cascade MCS.

In the second experiment, we test the performance of the proposed cascade MCS on the rest 20% testing documents. We use precision, recall, F1 for each class and overall accuracy for all the testing documents as performance measures which are defined as follows,

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}},$$
$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}},$$
$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}},$$
$$\text{overall accuracy} = \frac{\text{number of correctly classified documents}}{\text{number of overall documents}}.$$

To validate the merit of our proposed cascade MCS, we compare the experimental results against individual Classifier 1 and Classifier 2 where the number of candidate labels from Classifier 1 is fixed to one. Fig. 5 plots the F1, precision and recall measures for all 27 classes. By comparing Classifier 1 and Classifier 2, we can clearly see that the hierarchical classification model based on textual features outperforms Classifier 1 with image features. This reinforces our claim that textual features work much better than image features in document

Table 1. Overall Accuracy

	Cascade MCS	Classifier 1	Classifier 2
Overall Accuracy	0.925	0.452	0.874

image classification because documents with similar appearance might belong to different classes due to different textual information. On the other hand, by combining these two classifiers in the proposed MCS architecture, it improves the classification performance. The proposed cascade MCS exhibits better F1, precision and recall than any of individual classifiers. It achieves 5% increase in the overall accuracy over Classifier 2 as shown in Table 1. These results show the strength of the proposed cascade multiple classifier system in document image classification.

4 Conclusion

We have presented a cascade multiple classifier system for single-label multi-class document image classification problem in the paper. Two classifiers with different feature sets and architectures are combined in a cascade approach. The first classifier uses image features of documents to provide a set of fix number of candidate labels. The second classifier is a hierarchical classification model based on the textual features. It is composed of multiple RLS and KNN classifiers by exploiting the intrinsic hierarchies of the document class. By considering the candidate labels set from the proceeding classifier, the second classifier is able to search a much smaller region of the hierarchical tree and reaches a final decision. Experimental results on a large scale tax document image categorization suggest that the proposed method outperforms individual classifiers.

References

1. Chen, N., Blostein, D.: A survey of document image classification: problem statement, classifier architecture and performance evaluation. Int. J. Doc. Anal. Recognit. 10, 1–16 (2007)
2. Héroux, P., Diana, S., Ribert, A., Trupin, E.: Classification method study for automatic form class identification. In: Proc. Intl. Conf. on Pattern Recognition (ICPR), Brisbane, Australia, pp. 926–929 (1998)
3. Wenzel, C., Baumann, S., Jäger, T.: Advances in document classification by voting of competitive approaches. In: Proc. of Intl. Asso. for Pattern Recognition Workshop on Doc. Anal. Syst. (DAS), Malvern, USA, Octber 1996, pp. 352–372 (1996)
4. Alpaydin, E., Kaynak, C.: Cascading classifiers. Kybernetika 34, 369–374 (1998)
5. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. 20(3), 226–239 (1998)
6. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Trans. Syst., Man and Cybern. 22(3), 418–435 (1992)

7. Kittler, J., Matas, G., Jonsson, K., Sánchez, M.: Combining evidence in personal identity verification systems. Pattern Recog. Lett. 18(9), 845–852 (1997)
8. Huang, Y.S., Suen, C.Y.: A method of combining multiple experts for the recognition of unconstrained handwritten numerals. IEEE Trans. Pattern Anal. Mach. Intell. 17(1) (1995)
9. Woods, K., Kegelmeyer, W.P., Bowyer, K.: Combination of multiple classifiers using local accuracy estimates. IEEE Trans. Pattern Anal. Mach. Intell. 19(4), 405–410 (1997)
10. Larkey, L.S., Croft, W.B.: Combining classifiers in text categorization. In: Proc. of ACM SIGIR, pp. 289–297 (1996)
11. Hull, D., Pedersen, J., Schuetze, H.: Method combination for document filtering. In: Proc. of ACM SIGIR, pp. 279–287 (1996)
12. Yang, Y., Ault, T., Pierce, T.: Combining multiple learning strategies for effective cross validation. In: Proc. Intl. Conf. on Mach. Learn. (ICML), pp. 1167–1182 (2000)
13. Bennett, P.N., Dumais, S., Horvitz, E.: Probabilistic combination of text classifier using reliability indicators: Models and results. In: Proc. of ACM SIGIR, pp. 207–214 (2002)
14. Sarkar, P.: Image classification: classifying distributions of visual features. In: Proc. Intl. Conf. on Pattern Recognition (ICPR), Hong Kong, pp. 472–475 (2006)
15. Shin, C., Doermann, D., Rosenfeld, A.: Classification of document pages using structure-based features. Int. J. Doc. Anal. Recognit. 3(4), 232–247 (2001)
16. Xu, J., Singh, V., Govindaraju, V., Neogi, D.: A hierarchical classification model for document categorization. In: Proc. Intl. Conf. on Doc. Anal. Recognit (ICDAR), Barcelona, Spain (July 2009)

Maximum Membership Scale Selection

A Classifier Combining Approach to Multi-scale Image Segmentation

Marco Loog, Yan Li, and David M. J. Tax

Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
m.loog@tudelft.nl

Abstract. The use of multi-scale features is explored in the setting of supervised image segmentation by means of pixel classification. More specifically, we consider an interesting link between so-called scale selection and the maximum combination rule from pattern recognition. The parallel with scale selection is drawn further and a multi-scale segmentation method is introduced that relies on a per-scale classification followed by an over-scale fusion of these outcomes. A limited number of experiments is presented to provide some further understanding of the technique proposed.

1 Introduction

The fact that images are inherently multi-scale [1,2] has given rise to various paradigms within computer vision, image analysis and image processing that concur with this view. Indeed realizing that image features—whether these are edges, corners, blobs, ridges, or more complex structures [3,4,5,6]—typically occur at a particular scale, current detectors are developed to operate at multiple scales, employing a wavelet or scale space representation of the image [5,7,8]. Another fact is that such approaches have shown to improve upon raw-pixel-based approaches, providing a more complete picture of the actual features present and being less responsive to spurious structures. One of the most notable approaches is the one based on so-called scale selection, which suggests to detect a feature simultaneously with its optimal, intrinsic scale [9,10]. In a sense, this optimal scale is the scale at which this feature is most pronounced, most discernable from its surroundings (see also Section 2).

It should be pointed out that the lion's share of these methods are concerned with unsupervised techniques in which the feature to be detected is modeled using what is sometimes referred to as "geometric reasoning" [5] or the like [11,12]. In the supervised setting, say the task considered in this paper, i.e., image segmentation by means of supervised pixel classification [13,14,15,16], multi-scale information is typically incorporated by a simple concatenation of features[1] extracted from various scales into a

[1] N.B. Here "features" does not refer to an image feature like an edge, corner, or whatever, but in this case the "features" are those that serve as input for a classifier and are typically filter outputs, e.g., the output of a gradient operator or other edge filter. We do hope that it is clear from the context at what instance which type of "features" we mean.

single feature vector. Using such a representation, a classifier is subsequently built that takes this as input and tries to separate the one class, e.g., object, from the other, e.g., background, in an optimal way.

While smooth filter-based features are typically more powerful, as they are —apparently, and conceivably so—biased in the right direction, it leaves unsolved the issue of which scales to include. This is in addition to the question of what type of features to use in the first place of course. Leaving the latter concern aside for the time being, the investigation presented here focuses on the first issue and suggest a way to deal with it, or rather, to circumvent it.

Partly adopting the classical scale space thought that if one cannot decide on the particular scales to use a priori that one shall consider all scales simultaneously [1], we indeed set out to consider every scales in performing a supervised image segmentation task (in principle, that is). A second concept from the scale space community, which actually provides the chief inspiration for this work, is the earlier-mentioned scale selection approach [9,10]. It, first of all, triggered us to consider classifiers at individual scales for every pixel, in this avoiding the concatenation of arbitrary scales, but it also suggests how to afterward combine the posterior probabilities obtained for every scale into one single decision per pixel, namely by means of the maximum membership rule (or simply the max rule) [17,18].

The main points of this contribution are to demonstrate that an a priori scale selection can be avoided, to illustrate that improved performance over single scales can be attained through a simple fixed combining rule, and to provide some insight into when such performance increase may take place.

Outline. The next section, Section 2, starts out with a brief introduction of scale space theory in which many issues and topics for discussion are deliberately left aside as they do not pertain to the core of this contribution. After that, some more words are spent on the basics of automatic scale selection, which are subsequently related to supervised image segmentation by per-scale pixel classification and the maximum membership rule. Section 3 is reserved for a description of the experiments carried out and what we would like these to illustrate. The actual outcomes are contained in the same section. Section 4 concludes the paper and dwells upon some matters that were not considered in this work and sketches some topics of interest for future research.

2 Scale Space and Scale Selection

Scale Space Theory. Scale space theory can be motivated both from the viewpoint of physical measurements [19] and the theory on biological visual perception [1,20]. The theory provides a formal way to approach the problem of image representation for which it turns out that scale plays a crucial role. The most rudimentary incarnation of a scale space is so-called linear, or Gaussian, scale space, which relies on a physically plausible axiomatization [19]. This scale space is also the one we rely on.

In effect, given an image $\ell : \mathbb{R}^n \to \mathbb{R}$, the formal multi-scale image representation boils down to a one-parameter family of blurred images in which the blurring corresponds to a convolution with a Gaussian kernel which standard deviation σ is directly proportional to the scale at which the original image is represented. That is, a scale space representation $L : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}$ of ℓ is given by $L(x;\sigma) = (\ell * g_\sigma)(x)$, in which the Gaussian kernel g_σ is defined as

$$g_\sigma(x) = \frac{1}{(2\pi)^{\frac{n}{2}}\sigma^n} \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right).$$

From the previous definition, the scale space framework extends to a theory for Gaussian derivative operators in which general $|k|$th order derivatives $\frac{\partial^{|k|}}{\partial x^k} g_\sigma$ (k is a multi-index and $|k|$ is its order) are used to convolve the input image ℓ with [21,19,20]. A basic collection of features that can be calculates is the N-jet, which are all Gaussian image derivatives up to order N at a particular scale and which one may compare to the well-known idea of Taylor expansion. It is these basic features that are often employed in supervised image analysis techniques [13,14,15,16] and the order rarely goes beyond four (cf. [20]). The collection of basic linear filters, in turn, is used as a basis for expressing a large class of more general, potentially nonlinear, image processing operators [3,4,5,9,10]. One of the most important consequences from a more practical point is that many classical operators can now also be made to operate at various scale, as such enhancing their applicability (see for instance [6]). Moreover, it allows image operators to be made scale invariant, which is beneficial in many computer vision tasks.

Scale Selection. Although the original scale space contributions recognized that an image feature typically reside at a particular scale, they did not address the problem of how to select the locally appropriate scale to perform a task. The general scale selection framework developed in [9] and [10] provides a solution to this problem by generating hypotheses about interesting scale levels in an image based on a general principle stating that local extrema over scales in the operator output are likely candidates to correspond to interesting structures. In other words, an image filter is applied to all scales and a special meaning is attached to those scales at which the filter output attains either a local maximum or minimum.

Typically, the global maximum of the absolute output is taken to be the single optimal scale to detect the relevant feature. A simple interpretation of this idea is that the scale is selected at which the feature's presence is most pronounced or best matches the hypothesis. For a more rigorous and complete treatment of the scale selection principle, including the important notion of scale normalization, we refer to [9] and [10]. The outline above should suffice, however, to understand the gist of this paper.

Supervised Scale Selection. Assuming that both image data and label images are at our disposal, the supervised image segmentation approach we suggest is as follows. In the training phase, for every image location and every scale, features (Gaussian derivative filter outputs) are extracted. In this, the type of features remains fixed, only the scale varies. In the experiments we use for instance simple N-jets of a certain order at every scale. Following the feature extraction, a classifier is trained for every scale, i.e., using the corresponding features coming from that scale a mapping is learned that maps every feature vector to a class label or, in our case, a set of posteriors.

Now, in the testing phase, the same features are extracted and the proper classifier is applied at every scale, resulting in a posterior image for every scale. This in fact corresponds to regular image filtering, like in classical scale space approaches, only now the filter outputs are relatively complicated, consisting of posteriors coming from trained classifiers.

After the classification has been performed at every scale using the corresponding features, to come to an actual image label for every image location, the posterior information from all scales needs to be fused or combined over scales into a single decision for every image location. In order to do so, we suggest to proceed in the same scale selection spirit and find for every pixel the hypothesis, which supposes that the pixel belongs to one of several known classes, that gives the best match. In other words, the pixel should be labeled using the most confident classifier output, i.e., a maximum membership rule is employed.

It may be noted at this point that it may seem obvious to use other combination rules as well. The main problem with most of these rules, however, is the fact that they are generally sensitive to the way scale is parameterized. The maximum membership rule is one that does not have this drawback.

3 Experimental Investigations

In this section we report on several experiments based on different image data sets. Three of them are illustrated more extensively. Two of these are artificial and one of them is based on real image data coming from [16]. Some other experiments conducted were also meant to demonstrate some properties of our scheme that could be readily understood, i.e., based on a problem for which it was obvious that the combining scheme should outperform the single-scale classifiers. This initially turned out to be much harder than expected and at various occasions the results obtained were counterintuitive, at least at first. The next subsection reports briefly on some of these initial findings, while the other three subsections indeed provide some exemplifying tests.

Before moving on to the first subsection, we mention that in the latter three experiments all classifications were performed using quadratic discriminant analysis (QDA)—assuming normally distributed classes, all features extracted were N-jets (N equals 2, 5, and 3, respectively, the choices of which were based on small pilot studies), and in all cases we used 36 exponentially increasing scales ranging from $\frac{1}{2}$ to 24. (As hinted at earlier on, we of course take all scales into account only in principle. Solving an actual problem forces us to perform a discretization first and limit the range of scales.)

First Experiments. To start with, it should be mentioned that in non of our small initial experiments, we did not see any significant deterioration of the results after combining over all scales. Apparently, even though different scales were picked, the suggested approach at least managed to perform at a level similar to the optimal single scale.

Getting clear performance improvements however was more difficult. Our first attempts involved simple objects (e.g. white squares) of different sizes that should be segmented from their background (e.g. uniformly black), simple objects (e.g. white circles) blurred to variable levels against the same background, different levels of additive noise, and various combinations of these ingredients. The idea was that different object sizes or smoothing levels directly corresponds to varying scales, but all experiments basically failed to demonstrate that a potential improvement is possible.

In reconsideration, it may actually be quite obvious, or at least plausible, that with these simple images not much gain can come from the maximum membership rule. Take for example the noiseless case in which a close-to-optimal solution is basically a supervised thresholding of the gray values. There is really no need to perform operations at different scales. When adding some noise, the same applies, but possibly some blur is needed; adding noise does not suddenly make it needful to take into account various scales, although the best single scale usually increases slightly. All of this changes however when relatively severe noise is added to relatively large structures as the next paragraph illustrates.

Frequency Data. When the noise level is in some way proportional to the scale of the structure to be segmented (which are white on black, say) the following can happen. At small scales the small structures are well segmented because the noise is low and little blur can deal with this. Because of the large scale noise, the large scale structures cannot be segmented at this low scale. While small scale structures get removed under severe blur, large scale structures remain more intact in this situations and chances are that the smoothing takes care of the heavier noise and allows for a better segmentation of the larger objects.

This is tested by the experiment described on the basis of Fig. 1. The left image shows an input image, while the right image shows the ground truth. Other images in the database had the same structure: horizontal bands of two different frequencies, reflecting the large and small structures. The amplitude of the low frequency components was drastically reduced compared to the high frequency component such that the additive noise with constant variance had a detrimental impact on the visibility of the large structures. Four images were used for training, four for testing. Fig. 2 illustrates that small scales pick up the small, high frequency objects, while large scales mainly enable the correct classification of the low frequency part. Fig. 3 shows the improvements possible by applying the max rule, which over 10 test examples measures to about 40% reduction in classification error. The figures include some further data for which we refer to the respective captions.

Multi-Scale Texture Data. The second test is somewhat similar in spirit to the previous in the sense that we explicitly constructed a problem that needs at least two scales to be solved. We made two base textures, one in which dots are arranged quadrilaterally

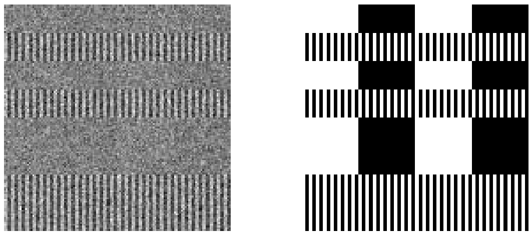


Fig. 1. Example input and output images from the frequency data, which is a two-class problem

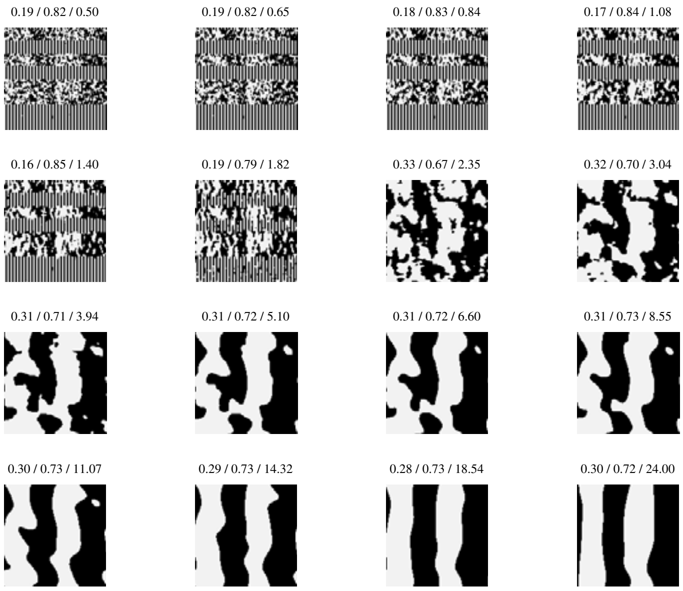


Fig. 2. Segmentation results at 16 different scales for the example image in Fig. 1. The left number in the titles of the images gives the error for that image, the middle one gives the average confidence, and the final number gives the feature scale.

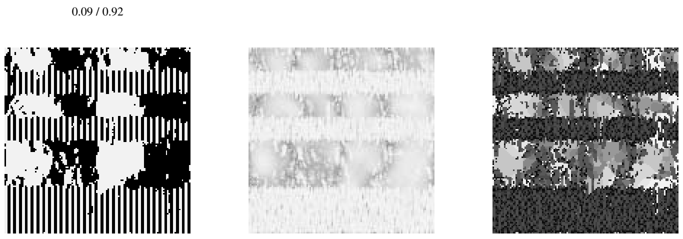


Fig. 3. Segmentation result of the maximum membership rule is on the left, in the middle is an image of the confidences at each location, and the right image gives the local scales (white equals 24, black $\frac{1}{2}$). The numbers in the title of the left image gives the error and the average confidence, respectively.

(::) and one in which they are arranged trilaterally (.˙.), which at high enough scale look basically the same. Two more textures were created that are simply scaled versions of the two base textures. By adding a large scale texture to a small scale, we can construct four different textures. We took three of these to construct two classes. One class contains two textures and the second only one. Most importantly, the textures are chosen such that the first texture from the first class can only be discriminated from the second class by means of the high scale texture, while the second can only be separated from the second class using the low scale texture. Fig. 4 shows an example input image and

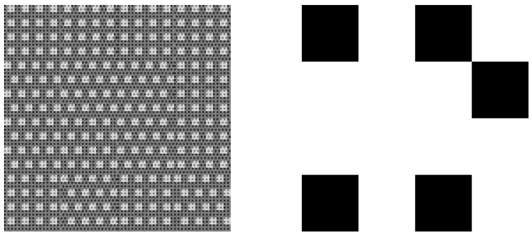


Fig. 4. Example input and output images from the texture data, which is a two-class problem

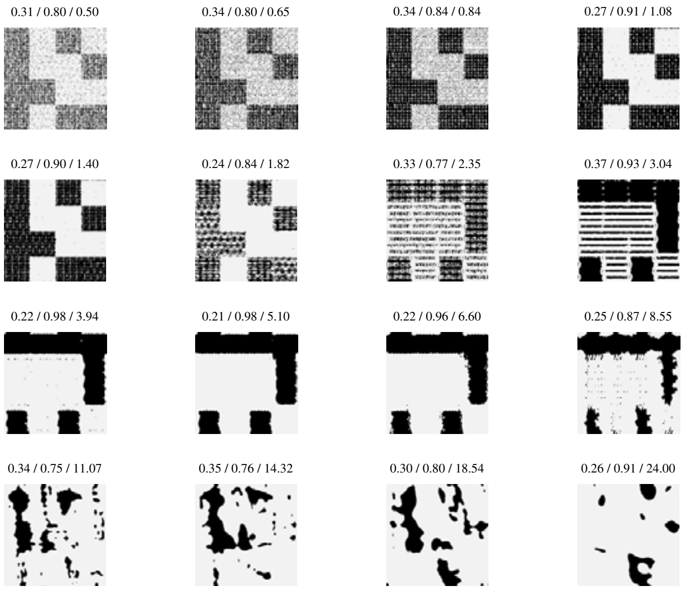


Fig. 5. Segmentation results at 16 different scales for the example image in Fig. 4. The left number in the titles of the images gives the error for that image, the middle one gives the average confidence, and the final number gives the feature scale.

the distribution of the two classes. Note that it is difficult to perceptually discriminate between the various textures (zooming in on the pdf file might help). In our tests, four images were used for training and four for testing.

Fig. 5 illustrates quite convincingly that indeed different separation takes place on essentially two different scales. Fig. 6 gives the outcome of the combiner, which clearly improves over the performance of individual classifiers. Again, on 10 previously unseen test image, an overall relative improvement of about 40% is attained.

Rib and Lung Segmentation. The final experiment is on data from [16]. It concerns a three-class rib and lung field segmentation task, see Fig. 7 for an example. This problem is mainly included to illustrate the procedure on a real-world data set. Looking at the

Fig. 6. Segmentation result of the maximum membership rule is on the left, in the middle is an image of the confidences at each location, and the right image gives the local scales (white equals 24, black $\frac{1}{2}$). The numbers in the title of the left image gives the error and the average confidence, respectively.

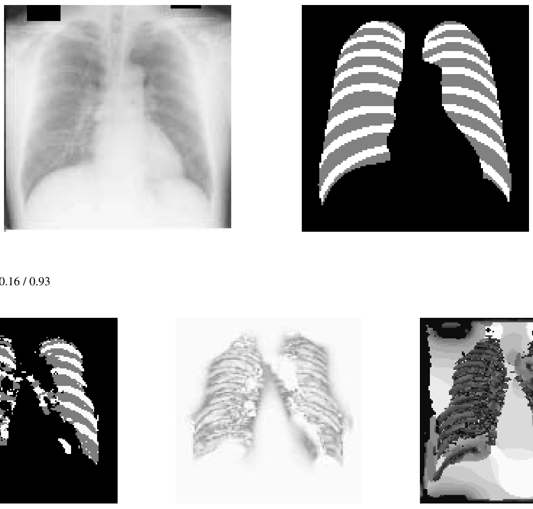


Fig. 7. Top row: Example input and output images from the lung and rib data, which is a three-class problem. Bottom row: Segmentation result of the maximum membership rule is on the left, in the middle is an image of the confidences at each location, and the right image gives the local scales (white equals 24, black $\frac{1}{2}$). The numbers in the title of the left image gives the error and the average confidence, respectively.

segmentations over scale (not included because of space limitations), it seems that the gain of combining comes from the fact that the lung fields (including the ribs) are better segmented from the background at a larger scale than the one needed to separate ribs from lungs. Even though this seems a plausible explanation, it should not be taken as conclusive as it is of a similar nature as the one provided in the first subsection above.

Nonetheless, training on ten images and testing on another ten, the combination approach does again provide improved performance and the overall error rate drops from about 0.24 for the optimal single scale classifier to around 0.18 for the max rule.

Fig. 8 gives a somewhat favorable example segmentation obtained with the combination scheme. The scale image illustrates that ribs are segmented on small scales while the lung field vs. background problem is solved on a higher scale.

4 Discussion and Conclusion

A link between scale selection from multi-scale image analysis, notably scale space theory, and a standard fixed classifier combination rule, the maximum membership rule, has been discussed. Based on the latter observation and some additional inspiration coming from the scale selection literature, a supervised image segmentation procedure was suggested that circumvents the normally encountered problem of choosing the scale. Its usefulness has been tested in a limited number of experiments and some insight was offered into the underlying mechanism.

Even though the parallels with classical scale selection are certainly of interest and the demonstrated performance of the approach is promising, we realize the paper offers only an initial study into the matter of combining multi-scale image processing technique with supervised learning methods. Even limiting ourselves to the current scope, there are many interrelationships not explored and intricacies not fully understood. Some of these are most surely interesting topics for future research.

Currently, for us the main intertwined issues are the interdependencies of various “scales”; the scale of the object or structure to be segmented, the scale of its blur, the scale of the noise, the scale of the extracted features, and the type of classifier, which could, in addition, has its own optimal scale to operate on—think for example of a Parzen classifier or even the QDA employed in this work. Besides these issues, there are of course the more standard ones, e.g. what types of features to use in the first place? What order of derivatives is sufficient? Do other classifier behave radically different in this setting? Would other combining rules not be more appropriate?

A laterally related question that has our attention is whether we can say anything useful on the way class distributions change under scale changes. Having knowledge on this might help to devise more powerful combination schemes.

Finally, when more and more complex tasks are considered, the issue may arises of how to combine scale information over different scales in a more structural kind of way. In order to solve this, we could consider to adopt the classical scale space thought in full and delve into the theory of deep structure [22,1,23,24]. This branch of scale space theory is, however, still in its infancy, even for the unsupervised case.

Acknowledgements. Marco Loog is financially supported by the Innovational Research Incentives Scheme of the Netherlands Research Organization [NWO], the Netherlands.

References

1. Koenderink, J.: The structure of images. Biological Cybernetics 50(5), 363–370 (1984)
2. Witkin, A.: Scale-space filtering: A new approach to multi-scale description. IEEE International Conference on Acoustics, Speech and Signal Processing 9 (1984)

3. Chen, C., Lee, J., Sun, Y.: Wavelet transformation for gray-level corner detection. Pattern Recognition 28(6), 853–861 (1995)
4. Deriche, R., Giraudon, G.: A computational approach for corner and vertex detection. International Journal of Computer Vision 10(2), 101–124 (1993)
5. ter Haar Romeny, B.: Front-end vision and multi-scale image analysis. Kluwer Academic, Dordrecht (2002)
6. Mikolajczyk, K., Schmid, C.: Scale & Affine Invariant Interest Point Detectors. International Journal of Computer Vision 60(1), 63–86 (2004)
7. Lindeberg, T.: Scale-Space Theory in Computer Vision. Kluwer Academic, Dordrecht (1994)
8. Mallat, S.: A Wavelet Tour of Signal Processing. Academic Press, London (1999)
9. Lindeberg, T.: Feature detection with automatic scale selection. International Journal of Computer Vision 30(2), 79–116 (1998)
10. Lindeberg, T.: Edge detection and ridge detection with automatic scale selection. International Journal of Computer Vision 30(2), 117–154 (1998)
11. Forstner, W., Gulch, E.: A fast operator for detection and precise location of distinct points, corners and centres of circular features. In: Proceedings of the ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data, pp. 281–305 (1987)
12. Rohr, K.: Recognizing corners by fitting parametric models. International Journal of Computer Vision 9(3), 213–230 (1992)
13. de Bruijne, M.: Shape particle guided tissue classification. In: IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (2006)
14. Folkesson, J., Dam, E., Olsen, O., Pettersen, P., Christiansen, C.: Segmenting articular cartilage automatically using a voxel classification approach. IEEE Transactions on Medical Imaging 26(1), 106–115 (2007)
15. van Ginneken, B., Stegmann, M., Loog, M.: Segmentation of anatomical structures in chest radiographs using supervised methods: a comparative study on a public database. Medical Image Analysis 10(1), 19–40 (2006)
16. Loog, M., Ginneken, B.: Segmentation of the posterior ribs in chest radiographs using iterated contextual pixel classification. IEEE Transactions on Medical Imaging 25(5), 602–611 (2006)
17. Kittler, J.: Combining classifiers: A theoretical framework. Pattern Analysis & Applications 1(1), 18–27 (1998)
18. Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
19. Florack, L.: Image Structure. Kluwer Academic Publishers, Dordrecht (1997)
20. Koenderink, J., van Doorn, A.: Receptive field families. Biological Cybernetics 63(4), 291–297 (1990)
21. Florack, L., ter Haar Romeny, B., Koenderink, J., Viergever, M.: Scale and the differential structure of images. Image and Vision Computing 10(6), 376–388 (1992)
22. Florack, L., Kuijper, A.: The Topological Structure of Scale-Space Images. Journal of Mathematical Imaging and Vision 12(1), 65–79 (2000)
23. Kuijper, A., Florack, L.: The hierarchical structure of images. IEEE Transactions on Image Processing 12(9), 1067–1079 (2003)
24. Olsen, O., Florack, L., Kuijper, A. (eds.): DSSCV 2005. LNCS, vol. 3753. Springer, Heidelberg (2005)

An Empirical Study of a Linear Regression Combiner on Multi-class Data Sets

Chun-Xia Zhang[1,2] and Robert P.W. Duin[2]

[1] School of Science and State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an Shaanxi 710049, China
cxzhang@mail.xjtu.edu.cn

[2] Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
r.duin@ieee.org

Abstract. The meta-learner *MLR* (Multi-response Linear Regression) has been proposed as a trainable combiner for fusing heterogeneous base-level classifiers. Although it has interesting properties, it never has been evaluated extensively up to now. This paper employs learning curves to investigate the relative performance of *MLR* for solving multi-class classification problems in comparison with other trainable combiners. Several strategies (namely, *Reusing*, *Validation* and *Stacking*) are considered for using the available data to train both the base-level classifiers and the combiner. Experimental results show that due to the limited complexity of *MLR*, it can outperform the other combiners for small sample sizes when the *Validation* or *Stacking* strategy is adopted. Therefore, *MLR* should be a preferential choice of trainable combiners when solving a multi-class task with small sample size.

Keywords: Ensemble classifier, Multi-response linear regression (*MLR*), Trainable combiner, Decision template (*DT*), Fisher linear discriminant (*FLD*).

1 Introduction

The task of constructing an ensemble classifier [1,2] can be broken into two steps, that is, generating a diverse set of base-level classifiers and combining their predictions. In general, there are two commonly used approaches to generate multiple base-level classifiers. One method is to train classifiers from different executions of the same learning algorithm through injecting randomness into the learning algorithm or manipulating the training examples, the input attributes and the outputs [3,4,5]. The obtained classifiers are often called *homogeneous* and they are typically combined by fixed combination rules such as weighted or unweighted voting. The other method is to apply some different learning algorithms to the same data set [6,7,8,9,10,11,12]. The derived classifiers are called *heterogeneous* and trainable combiners can generally merge them to result in a good ensemble classifier [7]. In the present study, we will focus on the heterogeneous base-level classifiers and trainable combiners.

With respect to trainable combiners, we are faced with the following problem: how to utilize the available data to train the models for both levels, the base-level classifiers as well as the combiner? Thus far, three strategies (that is, *Reusing*, *Validation* and *Stacking*) [13] have been proposed and they will be briefly described in Section 2.

In recent years, multi-response linear regression (*MLR*) has been recommended as a trainable combiner for merging heterogeneous base-level classifiers and there have been some variants of it [6,7,9,10,14]. Among the different methods related to *MLR*, the approach proposed in [7] may be the prominent one and some evidence has shown that it can handle multi-class problems very well [9]. To the best of our knowledge, however, the previous researchers only considered the situation that the training set size is supposed to be fixed and the *Stacking* method is employed to construct its meta-level data (namely, the data for training the combiner).

It is still unclear whether *MLR* will always keep its superiority with respect to different sample sizes and using different techniques to form its meta-level data. Thus, in this paper we employ learning curves to investigate the relative performance of *MLR* for solving multi-class classification problems in comparison with other combiners *FLD* (Fisher Linear Discriminant), *DT* (Decision Template) and *MEAN*. Several strategies are considered for using the available data to train the base-level classifiers and the combiner. The experimental results show that for small sample sizes, *MLR* can generally outperform the other combiners when the *Validation* or *Stacking* strategy is adopted. Meanwhile, the *Reusing* strategy should be avoided as much as possible anyway. When the sample size is large, however, there is little difference between the compared combiners no matter what strategy is employed to form the meta-level data.

The remainder of the paper is organized as follows. In Section 2, the working mechanism of *MLR* as well as some feasible strategies to utilize the given data to derive both base-level classifiers and combiner is introduced. Section 3 presents and discusses the experimental studies conducted on some multi-class data sets. Finally, the main conclusions are given in Section 4.

2 *MLR* and Strategies to Use the Training Data

2.1 Working Mechanism of *MLR*

Consider a given set $\mathscr{L} = \{(y_n, \mathbf{x}_n)\}_{n=1}^{N}$, where y_n is a class label taking value from $\Phi = \{\omega_1, \omega_2, \cdots, \omega_m\}$ and $\mathbf{x}_n \in R^d$ is a vector representing the attribute values of the nth example. Suppose that a set $\mathscr{C} = \{C_1, C_2, \cdots, C_L\}$ of L base-level classifiers is generated by applying the heterogeneous learning algorithms $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_L$ to $\mathscr{L}$ and the prediction of $C_i (i = 1, 2, \cdots, L)$ when applied to an example $\mathbf{x}$ is a probability distribution vector

$$\begin{aligned}\mathbf{P}_{C_i}(\mathbf{x}) &= (P_{C_i}(\omega_1|\mathbf{x}), P_{C_i}(\omega_2|\mathbf{x}), \cdots, P_{C_i}(\omega_m|\mathbf{x}))^T \\ &\triangleq (P_1^i(\mathbf{x}), P_2^i(\mathbf{x}), \cdots, P_m^i(\mathbf{x}))^T,\ i = 1, 2, \cdots, L,\end{aligned} \tag{1}$$

where $P_j^i(\mathbf{x})$ denotes the probability that the example $\mathbf{x}$ belongs to class ω_j as estimated by the classifier C_i. Furthermore, we define $\mathbf{P}(\mathbf{x})$ as an mL-dimensional column vector, namely,

$$\begin{aligned}\mathbf{P}(\mathbf{x}) &= (\mathbf{P}_1^T(\mathbf{x}), \mathbf{P}_2^T(\mathbf{x}), \cdots, \mathbf{P}_L^T(\mathbf{x}))^T \\ &= (\underbrace{P_1^1(\mathbf{x}), \cdots, P_m^1(\mathbf{x})}_{\text{Classifier } C_1}, \underbrace{P_1^2(\mathbf{x}), \cdots, P_m^2(\mathbf{x})}_{\text{Classifier } C_2}, \cdots, \underbrace{P_1^L(\mathbf{x}), \cdots, P_m^L(\mathbf{x})}_{\text{Classifier } C_L})^T \end{aligned} \quad (2)$$

Based on the intermediate feature space constituted by the outputs of each base-level classifier, the *MLR* method [7] firstly transforms the original classification task with m classes into m regression problems: the problem for class ω_j has examples with responses equal to one when they indeed have class label ω_j and zero otherwise. For each class ω_j, *MLR* selects only $P_j^1(\mathbf{x}), P_j^2(\mathbf{x}), \cdots, P_j^L(\mathbf{x})$, the probabilities that $\mathbf{x}$ belongs to ω_j predicted by the base-level classifiers $C_1, C_2, \cdots, C_L$, as the input attributes to establish a linear equation

$$LR_j(\mathbf{x}) = \sum_{i=1}^{L} \alpha_j^i P_j^i(\mathbf{x}),\ j = 1, 2, \cdots, m, \quad (3)$$

where the coefficients $\{\alpha_j^i\}_{i=1}^L$ are constraint to be non-negative and the non-negative-coefficient least-squares algorithm described in [15] is employed to estimate them. When classifying a new example $\mathbf{x}$, we only need to compute $LR_j(\mathbf{x})$ for all the m classes and assign it to the class ω_k which has the greatest value.

2.2 Strategies to Use the Training Data

In order to estimate the coefficients $\{\alpha_j^i\}_{i=1}^L$ in formula (3), or more generally, to train the combiner *MLR*, we have to form the meta-level data. In practical applications, however, we are only given a single set $\mathscr{L}$ which should be used to train the base-level classifiers as well as the combiner. In this situation, there may be three feasible approaches(*Reusing*, *Validation* and *Stacking*) to make full use of $\mathscr{L}$ to construct an ensemble classifier.

The *Reusing* strategy simply applies the given learning algorithms $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_L$ to $\mathscr{L}$ to train the base-level classifiers $C_1, C_2, \cdots, C_L$ which are then used to predict the examples in $\mathscr{L}$ to form the meta-level data. Since the same set $\mathscr{L}$ is used to derive both base-level classifiers and combiner, the obtained combiner will inevitably be biased.

The *Validation* strategy splits the training set $\mathscr{L}$ into two disjoint subsets, one of which is used to derive the base-level classifiers $C_1, C_2, \cdots, C_L$ and the other one is employed to construct the meta-level data.

The *Stacking* strategy utilizes the cross-validation method to form the meta-level data. Firstly, the training set $\mathscr{L}$ is partitioned into K disjoint subsets $\mathscr{L}_1, \mathscr{L}_2, \cdots, \mathscr{L}_K$ of almost equal size. In order to obtain the base-level predictions on examples in $\mathscr{L}_k$, say, $\mathscr{L}_k' = \{(y_i, \mathbf{P}^T(\mathbf{x}_i)) | (y_i, \mathbf{x}_i) \in \mathscr{L}_k\}$, the learning algorithms $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_L$ are applied to the set $\mathscr{L} \backslash \mathscr{L}_k$ to derive the classifiers $\{C_{j,k}\}_{j=1}^L$ which are then used to predict the examples in $\mathscr{L}_k$. After repeating this process K times (once for each set $\mathscr{L}_k$), we can obtain the final meta-level data set $\mathscr{L}^{CV} = \bigcup_{k=1}^{K} \mathscr{L}'_k$. The readers can refer to [2,6,7,13] for more details about this technique.

It is worthwhile to mention that for a new example $\mathbf{x}$, there may be two different ways to construct the input $\mathbf{P}(\mathbf{x})$ of the combiner trained by the *Stacking* method. The *Stacking I method*, commonly utilized by many researchers [16,6,7,9,10], is to retrain the base-level classifiers $C_1, C_2, \cdots, C_L$ on the full training set $\mathscr{L}$ to produce $\mathbf{P}(\mathbf{x})$. The *Stacking II method*, proposed in [13], applies all the classifiers $\{C_{j,k}\}_{j=1}^{L}{}_{k=1}^{K}$ which are trained in the process of cross-validation to predict $\mathbf{x}$. For each type of base classifier, it averages the predictions that are obtained in each fold, namely, $\bar{C}_j(\mathbf{x}) = (1/K)\sum_{k=1}^{K} C_{j,k}(\mathbf{x})$ and forms the input of the combiner as $\mathbf{P}(\mathbf{x}) = (\bar{C}_1^T(\mathbf{x}), \bar{C}_2^T(\mathbf{x}), \cdots, \bar{C}_L^T(\mathbf{x}))^T$.

3 Experimental Studies

Because learning curves can give a good picture to study the performance of an algorithm at various sample sizes, in this section we employ them to investigate the relative performance of *MLR* for solving multi-class classification tasks in comparison with several other combiners. Furthermore, each of the different strategies described in subsection 2.2 will be considered here to employ the given set to train the models for both levels.

3.1 Experimental Setup

We conducted experiments on a collection of 10 multi-class data sets from the UCI repository [17]. The data sets and some of their characteristics are summarized in Table 1.

Table 1. Summary of the data sets used in the experiments

Data set	# Examples	# Attributes	# Classes	Training size (Per class)	Test size (Per class)
Abalone	4177	10	3	5,10,20,30,50,80,100	500
Cbands	12000	30	24	10,20,30,50,80,100	100
Digits	2000	240	10	5,10,20,30,50,80,100	50
Letter	20000	16	26	10,20,30,50,80,100	200
Pendigits	7494/3498	16	10	5,10,20,30,50,80,100	3498(total)
Satellite	6435	36	6	5,10,20,30,50,80,100	500
Segmentation	2310	19	7	5,10,20,30,50,80,100	100
Vehicle	846	18	4	5,10,20,30,50,80,100	50
Vowelc	990	12	11	10,20,30,50	30
Waveform	5000	21	3	5,10,20,30,50,80,100	500

We totally considered 7 different types of base-level classifiers: Fisher linear discriminant (*fisherc*), Parzen density classifier (*parzenc*), Nearest neighbor (*knnc*), Logistic linear classifier (*loglc*), Nearest mean (*nmc*), Decision tree (*treec*), Support vector classifier (*svc*) and 4 different combiners: Multi-response linear regression (*MLR*), Fisher linear discriminant (*FLD*), Decision template

(DT) and Mean ($MEAN$), which were all implemented with PRTools[1]. The outputs of each base-level classifier were scaled to fall into a [0,1] interval so that the intermediate feature space is constituted in a homogeneous way. Considering the fact that we were combining the base-level classifiers with posterior probabilities or confidences as their outputs, the voting based combiners were not included. Furthermore, the parameters included into the base-level classifiers and the combiners were all taken to be their default values in PRTools. With the given base-level classifiers, two batches of experiments (one uses the first four ones and the other considers all the seven ones) were conducted to study the influence of the number of base-level classifiers on the relative performance of the combiners. The different strategies described in subsection 2.2 were respectively considered to utilize the available data to train base-level classifiers and combiners, in which the *Validation* method uses a 50%/50% split and the two *Stacking* methods employ 10-fold cross-validation to form the meta-level data.

We estimated the learning curves in the following way to understand the behavior of different combiners at various sample sizes. For each data set (except for "Pendigits") listed in Table 1, a training set and an independent test set with desired sizes were randomly sampled. As for the "Pendigits" data set which has separate training and testing data, all of its testing data was used as the test set. On each of the obtained training set, the base-level classifiers and the combiner were constructed according to each of the strategies described in subsection 2.2. The trained combiner was then executed on the test set and the estimated classification error was utilized to evaluate its performance. It should be noted that all steps required for building the base-level classifiers and the combiner, including the cross-validation utilized by the *Stacking* method, were performed on the training set only. For each training set size, the above process was repeated 10 times and the obtained results were averaged.

3.2 Results and Discussion

We firstly considered the cases that an ensemble classifiers is composed of 4 base-level classifiers. For each combiner, the mean of the test errors over 10 replications (standard deviation also shown) was plotted as a function of the sample size. Due to the limited space of this paper, only some representative plots obtained on "Cbands", "Digits" and "Satellite" data sets are presented here but the other ones are available from the authors. From these plots, the following observations can be made:

- For small sample sizes, *MLR* can generally perform better than the other combiners when the *Validation* or *Stacking* method is adopted. However, the superiority of *MLR* is not very obvious when the classification task has many classes ("Cbands") or high dimensionality of the input space ("Digits"), which may be caused by the inadequate diversity among the linear models established by *MLR*.

[1] PRTools is a Matlab Toolbox for Pattern Recognition and it can be freely downloaded from the PRTools website, http://www.prtools.org.

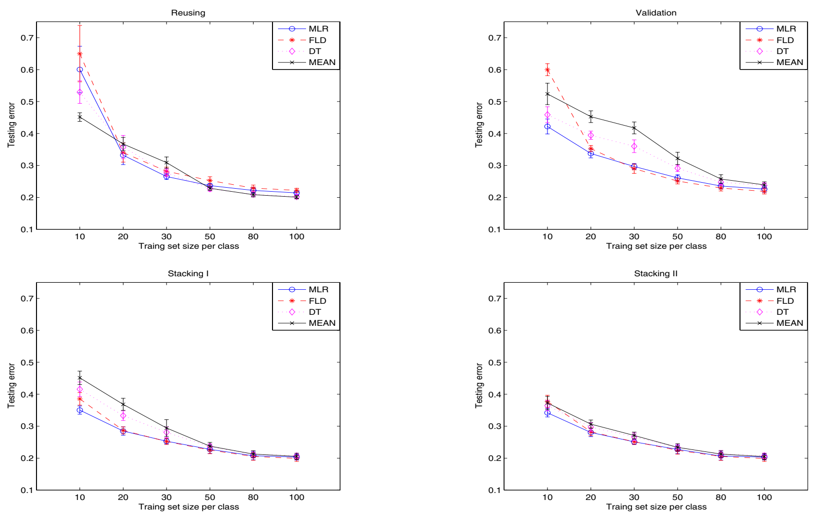


Fig. 1. On "**Cbands**" data set, learning curves for the compared combiners when different strategies are utilized to train base-level classifiers and combiner

- When the sample size is large, there is little difference between the compared combiners no matter what strategy is used to form the meta-level data.
- The *Reusing* strategy should be avoided as much as possible anyway except for large sample sizes because the test errors corresponding to it are generally larger than those obtained by using the *Validation* or *Stacking* technique.
- Some combiners are observed in Fig. 2 to exhibit a dramatic error in some situations, which can be explained as the bad performance of base-level classifier *fisherc* or combiner *FLD* when the training set size is comparable to the dimension of the feature space [13,18].
- For each combiner, the results obtained by *Stacking II* are slightly better than those derived by *Stacking I*. The reasons for this may be due to the fact that *Stacking II* benefits from more robust base-level classifiers [13].

In order to see clearly the relative performance of the compared combiners on the used 10 data sets, Table 2 provides some comparative summaries of the mean test errors for the combiner *MLR* in comparison with other ones. Here, the geometric means of error ratios and significant Wins-Losses of *MLR* compared with other combiners at each considered sample size were listed and these statistics were computed in the following way. Take the notation "*MLR/FLD*" as an example, at each sample size we firstly computed the error ratio of *MLR* to *FLD* on each data set, and then calculated the geometric mean of the obtained error ratios across all the data sets. Therefore, the value smaller than 1 indicates the better performance of *MLR*. With respect to the Win-Loss statistic, a paired t-test

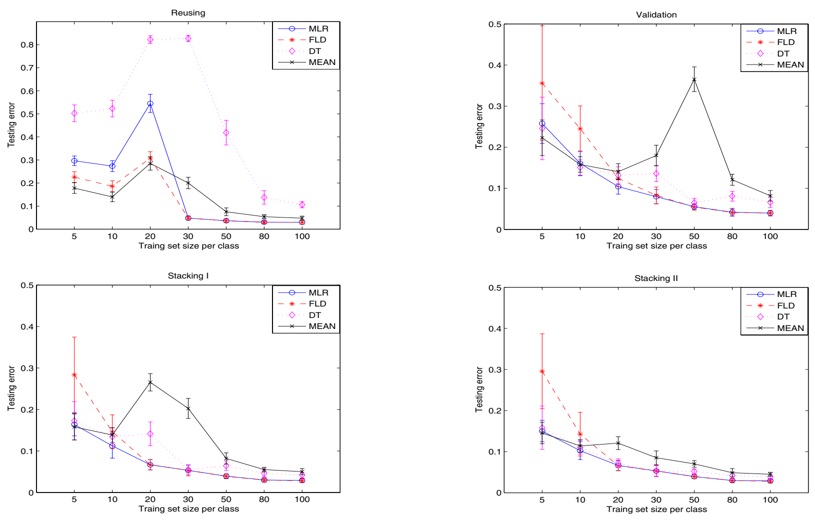


Fig. 2. On "**Digits**" data set, learning curves for the compared combiners when different strategies are utilized to train base-level classifiers and combiner

was utilized to check whether the performance of *MLR* is significantly better than that of the other ones at the significance level 0.05 for each combination of data set and sample size and the numbers listed in the table should be read as the number of data sets on which *MLR* performs significantly better than *FLD*. From the results reported in Table 2, we can draw almost the same conclusions as those obtained from the previous figures.

Table 2. On the used 10 data sets, geometric means of error ratios as well as significant Wins-Losses of *MLR* in comparison with other combiners (4 base-level classifiers)

		Training set size per class						
		5	10	20	30	50	80	100
Reusing method	*MLR/FLD*	0.9842	0.9898	1.0269	0.9757	0.9760	0.9952	0.9886
		3-2	5-1	4-1	4-0	2-0	4-1	3-1
	MLR/DT	1.0116	1.0575	1.0532	0.7557	0.7810	0.8404	0.8840
		1-3	1-6	2-4	3-5	2-4	3-5	3-5
	MLR/MEAN	1.2563	1.2221	1.1510	0.8289	0.9123	0.9066	0.9380
		0-6	0-8	1-5	4-2	2-4	3-5	4-5
Validation method	*MLR/FLD*	0.7045	0.7201	0.9517	1.0305	1.0367	1.0318	1.0371
		6-0	9-0	5-2	1-2	1-5	1-4	0-3
	MLR/DT	0.8454	0.8718	0.8897	0.8717	0.9150	0.8721	0.9004
		4-0	9-0	5-0	5-2	5-1	5-0	4-1
	MLR/MEAN	0.9051	0.8391	0.7779	0.7496	0.7070	0.7909	0.8326
		3-0	7-1	7-0	8-0	7-0	5-1	5-1
Stacking I method	*MLR/FLD*	0.6983	0.8535	0.9782	0.9908	1.0158	1.0211	1.0316
		7-0	10-0	2-1	1-1	1-4	0-4	0-4
	MLR/DT	0.8290	0.8569	0.8495	0.9239	0.9162	0.9037	0.9123
		5-0	6-0	5-0	4-1	4-0	4-1	4-1
	MLR/MEAN	0.8922	0.8091	0.7500	0.7622	0.8525	0.8532	0.8620
		3-0	6-0	5-0	5-1	6-1	4-1	4-1
Stacking II method	*MLR/FLD*	0.6919	0.8709	0.9869	0.9963	1.0171	1.0253	1.0347
		7-0	10-0	2-1	1-1	0-3	0-3	0-5
	MLR/DT	0.8499	0.9041	0.9377	0.9579	0.9385	0.9213	0.9216
		4-0	5-0	4-0	4-1	6-1	5-0	4-0
	MLR/MEAN	0.9299	0.8684	0.8513	0.8728	0.8743	0.8821	0.8816
		3-2	4-1	4-1	4-1	6-1	4-1	4-1

Table 3. On the considered 8 data sets, geometric means of error ratios as well as significant Wins-Losses of *MLR* in comparison with other combiners (7 base-level classifiers)

		Training set size per class						
		5	10	20	30	50	80	100
Reusing method	*MLR/FLD*	1.0750	0.9784	0.9973	0.9503	0.9619	0.9722	0.9676
		2-3	4-1	2-1	4-1	3-0	1-0	3-0
	MLR/DT	1.2238	1.1786	1.1888	0.8278	0.8078	0.8219	0.8534
		0-5	1-6	1-5	1-4	3-3	3-3	4-3
	MLR/MEAN	1.4937	1.3714	1.4900	1.0298	0.9185	0.8889	0.8825
		0-7	1-7	1-5	1-3	3-3	4-3	3-3
Validation method	*MLR/FLD*	0.8298	0.5536	0.7442	0.8878	1.0258	1.0177	1.0102
		4-0	8-0	7-0	6-1	1-1	0-1	0-1
	MLR/DT	0.9410	0.9812	0.9786	0.9476	0.9140	0.8784	0.8663
		1-0	4-1	3-1	3-0	4-0	4-0	5-0
	MLR/MEAN	1.0349	0.9104	0.8831	0.8355	0.7993	0.8176	0.8145
		1-2	6-0	4-0	8-0	6-0	5-0	6-0
Stacking I method	*MLR/FLD*	0.5368	0.7356	0.9036	0.9452	1.0042	1.0343	1.0134
		7-0	8-0	7-0	3-0	1-1	0-2	0-2
	MLR/DT	0.9759	0.9915	0.9259	0.9025	0.8712	0.8441	0.8240
		2-1	2-1	4-0	3-0	5-1	5-1	5-0
	MLR/MEAN	1.0112	0.9481	0.8764	0.8508	0.8516	0.8110	0.7911
		0-1	4-0	5-0	7-0	5-0	5-0	6-0
Stacking II method	*MLR/FLD*	0.5146	0.7427	0.9111	0.9650	1.0365	1.0266	1.0382
		7-0	8-0	4-0	2-1	0-3	0-2	0-2
	MLR/DT	0.9498	0.9866	0.9547	0.9085	0.8755	0.8335	0.8267
		0-0	1-0	4-0	4-0	6-0	4-0	5-0
	MLR/MEAN	1.0231	0.9440	0.8795	0.8289	0.7653	0.8091	0.8042
		0-0	4-0	4-0	5-0	7-0	5-0	5-0

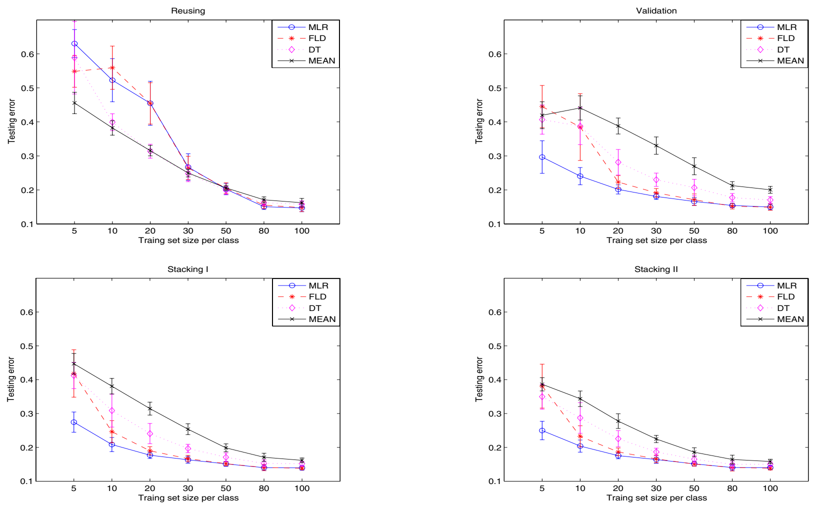


Fig. 3. On "**Satellite**" data set, learning curves for the compared combiners when different strategies are utilized to train base-level classifiers and combiner

As for the experiments conducted with the 7 base-level classifiers, we reported the obtained results in Table 3 whose format is identical to that of Table 2. Considering that it takes too much time to conduct experiments on "Cbands" and "Letter" data sets, only the remaining 8 ones were taken into account in this batch of experiments. Through comparing the results listed in Tables 2 and 3, it seems that the superiority of *MLR* over the other combiners is more significant when more base-classifiers are used to form an ensemble classifier since *MLR* is observed to loss less times in this case.

3.3 Complexity Analysis

Now let us analyze the complexity of each considered combiner in terms of the number of parameters to be estimated.

Apparently, the combiner *MEAN* have not any parameters to estimate since it simply averages the probability distributions predicted by each base-level classifier and assigns $\mathbf{x}$ to the class having the largest probability. Because the combiner *MLR* needs to establish m linear models and each of them has L parameters, there are totally mL parameters required to be estimated.

If we use $\overline{\mathbf{P}}^k = (P_{k,1}^1, P_{k,2}^1, \cdots, P_{k,m}^1, P_{k,1}^2, P_{k,2}^2, \cdots, P_{k,m}^2, \cdots, P_{k,1}^L, P_{k,2}^L, \cdots, P_{k,m}^L)^T$ to denote the mean vector of class ω_k in the intermediate feature space, the combiner *DT* estimates the class label of $\mathbf{x}$ as

$$\omega_{dt}(\mathbf{x}) = \underset{1\le k\le m}{\operatorname{argmin}} \sum_{i=1}^{L}\sum_{j=1}^{m}[P_{k,j}^i - P_j^i(\mathbf{x})]^2 = \underset{1\le k\le m}{\operatorname{argmin}} ||\overline{\mathbf{P}}^k - \mathbf{P}(\mathbf{x})||^2. \tag{4}$$

Thus, there are m^2L parameters to be estimated.

As for the combiner *FLD*, it decides the class label of $\mathbf{x}$ as

$$\omega_{fld}(\mathbf{x}) = \underset{1\le k\le m}{\operatorname{argmin}} (\overline{\mathbf{P}}^k - \mathbf{P}(\mathbf{x}))^T \Sigma^{-1} (\overline{\mathbf{P}}^k - \mathbf{P}(\mathbf{x})), \tag{5}$$

here Σ indicates the sample estimate of the $mL \times mL$ covariance matrix supposed to be common for each class. Since the combiner *FLD* utilizes one-against-all strategy to solve a multi-class task, it needs to estimate $m(\frac{mL(mL+1)}{2} + 2mL)$ parameters in total.

Based on the above analysis, we can see that the compared combiners can be ordered from simple to complex as *MEAN*, *MLR*, *DT* and *FLD*. Thus, one of the reasons for the better performance of the combiner *MLR* than that of *DT* and *FLD* at small sample sizes may be attributed to its limited complexity. As for the advantage of *MLR* over *MEAN*, it may be due to the fact that *MLR* takes into account more information in the data whereas *MEAN* does not.

4 Conclusions

In this paper, we utilized learning curves to investigate the relative performance of *MLR* for solving multi-class problems in comparison of other trainable combiners *FLD*, *DT* and the fixed combiner *MEAN*. The *Reusing*, *Validation* and two versions of *Stacking* method were respectively considered for using the given data to train the base-level classifiers as well as the combiner. The experimental results show that *MLR* can outperform the other combiners for small sample sizes when *Validation* or *Stacking* method is employed. Meanwhile, the *Reusing* strategy should be avoided as much as possible anyway. When the sample size is large, however, there is little difference between the compared combiners no matter what strategy is employed to form the meta-level data. As for the two

Stacking methods, *Stacking II* may be preferred over *Stacking I* for its robustness and relatively smaller computational cost.

Acknowledgements. This research was supported in part by China State Scholarship Fund, National Basic Research Program of China (973 Program) (Grant No. 2007CB311002) and National Natural Science Foundations of China (Grant Nos. 60675013 and 10531030). This work was conducted when the first author was in Pattern Recognition Group, Delft University of Technology.

References

1. Kuncheva, L.I., Bezdek, J.C., Duin, R.P.W.: Decision templates for multiple classifier fusion: an experimental comparison. Pattern Recog. 34(2), 299–314 (2001)
2. Todorovski, L., Džeroski, S.: Combining classifiers with meta decision trees. Mach. Learn. 50(3), 223–249 (2003)
3. Breiman, L.: Bagging predictors. Mach. Learn. 24(2), 123–140 (1996)
4. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: 13th International Conference on Machine Learning, pp. 148–156. Morgan Kaufmann Press, San Francisco (1996)
5. Breiman, L.: Randomizing outputs to increase prediction accuracy. Mach. Learn. 40(3), 229–242 (2000)
6. Ting, K.M., Witten, I.H.: Stacking bagged and dagged models. In: 14th International Conference on Machine Learning, pp. 367–375. Morgan Kaufmann Press, San Francisco (1997)
7. Ting, K.M., Witten, I.H.: Issues in stacked generalization. J. Artif. Intell. Res. 10, 271–289 (1999)
8. Merz, C.J.: Using corresponding analysis to combine classifiers. Mach. Learn. 36(1/2), 33–58 (1999)
9. Seewald, A.K.: How to make stacking better and faster while also taking care of an unknown weakness. In: 19th International Conference on Machine learning, pp. 554–561. Morgan Kaufmann Press, San Francisco (2002)
10. Džeroski, S., Ženko, B.: Is combining classifiers with stacking better than selecting the best ones? Mach. Learn. 54(3), 255–273 (2004)
11. Raudys, S.: Trainable fusion rules: I. Large sample size case. Neural Networks 19(10), 1506–1516 (2006)
12. Raudys, S.: Trainable fusion rules: II. Small sample-size effects. Neural Networks 19(10), 1517–1527 (2006)
13. Paclík, P., Landgrebe, T.C.W., Tax, D.M.J., Duin, R.P.W.: On deriving the second-stage training set for trainable combiners. In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS 2005. LNCS, vol. 3541, pp. 136–146. Springer, Heidelberg (2005)
14. Liu, M., Yuan, B.Z., Chen, J.F., Miao, Z.j.: Does linear combination outperform the *k*-NN rule? In: 8th International Conference on Signal Processing, vol. 3. IEEE Press, Beijing (2006)
15. Lawson, C.J., Hanson, R.J.: Solving Least Squares Problems. SIAM Publications, Philadephia (1995)
16. Wolpert, D.H.: Stacked generalization. Neural Networks 5(2), 241–259 (1992)
17. UCI machine larning respository, `http://www.ics.uci.edu/~mlearn/MLRespository.html`
18. Lai, C.: Supervised classification and spatial dependency analysis in human cancer using high throughput data. Ph.D Thesis, Delft University of Technology (2008)

A Study of Random Linear Oracle Ensembles

Amir Ahmad and Gavin Brown

School of Computer Science, University of Manchester,
Manchester, M13 9PL, UK
{ahmada,gbrown}@cs.man.ac.uk

Abstract. Random Linear Oracle (RLO) ensembles of Naive Bayes classifiers show excellent performance [12]. In this paper, we investigate the reasons for the success of RLO ensembles. Our study suggests that the decomposition of most of the classes of the dataset into two subclasses for each class is the reason for the success of the RLO method. Our study leads to the development of a new output manipulation based ensemble method; *Random Subclasses* (RS). In the proposed method, we create new subclasses from each subset of data points that belongs to the same class using RLO framework and consider each subclass as a class of its own. The comparative study suggests that RS is similar to RLO method, whereas RS is statistically better than or similar to Bagging and AdaBoost.M1 for most of the datasets. The similar performance of RLO and RS suggest that the creation of local structures (subclasses) is the main reason for the success of RLO. The another conclusion of this study is that RLO is more useful for classifiers (linear classifiers etc.) that have limited flexibility in their class boundaries. These classifiers can not learn complex class boundaries. Creating subclasses makes new, easier to learn, class boundaries.

Keywords: Classifier Ensemble, Naive Bayes, Clusters, Subclasses.

1 Introduction

Ensembles are combination of multiple base models [9,13]. Final classification depends on the combined outputs of individual models. Classifier ensembles have shown to produce better results than single models, if the classifiers, in the ensembles, are accurate and diverse [9,13]. Ensembles perform better when base models are unstable; classifiers whose output undergoes significant changes in response to small changes in training data. Decision-trees and neural networks are unstable classifiers. Naive Bayes and other linear classifiers are generally stable [15,14] thus not best suited for ensemble methods such as Bagging [2] and Boosting [6]. However, RLO ensembles have shown very good performance with Naive Bayes classifiers [12]. In this paper, we study RLO ensembles and propose a new ensemble method *Random Subclasses* (RS) that is less computationally expensive and more flexible as compared to RLO.

The paper is organised as follows. In section 2, we discuss the subclasses strategy to improve the performance of simple classifiers. RLO is discussed in

Section 3. In section 4, we introduce Random Subclasses (RS) method. Results are discussed in section 5. Section 6 describes conclusion and future work.

2 Subclasses Methods

In this section we discuss the subclasses strategy to improve the performance of low variance classifiers. Low variance classifiers (linear classifiers, linear discriminate analysis etc.) have limited flexibility in learning class boundaries. They have difficulty in learning class boundaries for the classes that consist of loosely disconnected components (Fig. 1). One of the methods to solve this problem is to create new class boundaries that can be learnt easily by the classifier. This is achieved by splitting the original set of classes into subclasses. Class decomposition via clustering [14,15], supervised clustering [5] etc. are different methods to create subclasses. These new subclasses can improve the performance of linear classifiers [14,15,5]. Naive Bayes is a linear classifier. We discuss Naive Bayes classifiers in detail to understand the effect of new subclasses on Naive Bayes classifiers.

A Naive Bayes classifier is a simple probabilistic classifier however it produces good results [4,8]. Naive Bayes is a global classifiers; it uses all the available data for estimating probabilities. When each class of the dataset lies in different regions, it is not able to estimate class probabilities accurately. In other words, it fails to detect local class variations as single-dimensions projection of the data loses their spatial information [15]. Rish et al. [11] show that in cases when data points belong to same class spread in multiple regions, computing marginals of the data may result in significant loss of information.

Vilalta and Rish propose a method [15] to improve the performance of Naive Bayes. In this method, they transform the data by decomposing each class into many clusters. In other words, they introduce local structures in the global classifier. Probabilities are computed on these new clusters. These predefined local structures (clusters) help in computing better probability estimates. Naive Bayes classifiers give improved results with this additional step.

In the next section, we will discuss RLO that creates the same kind of local structures as discussed in the last two sections.

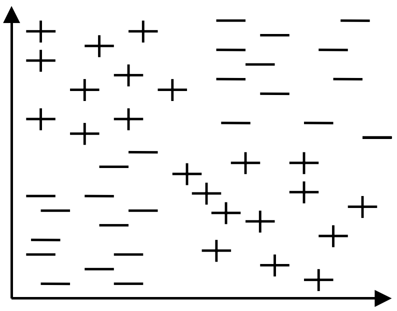


Fig. 1. Two class problem, each class has two loosely connected components

3 Random Linear Oracle

Kuncheva and Rodriguez [10,12] propose Random Linear Oracle (RLO) ensemble method. This algorithm is presented in Fig. 2. In this method, every classifier in the ensemble consists of a pair of classifiers. These two classifiers learn on different subspaces decided by the random hyperplane. While testing, first the position of the testing data point is decided by the random hyperplane then the decision of the classifier, which is trained in that subspace, is taken. They suggest two random oracles (random hyperplane); random linear oracle and random spherical oracle, however in this work, we focus on Random Linear Oracle ensembles. The similar discussion is true for Random Spherical Oracle ensembles.

In the RLO approach, the space is divided into two subspaces using a hyperplane drawn in the feature space of the dataset. The hyperplane is generated by taking two random points from the training set and calculating the hyperplane perpendicular to the line segment between the points and running through the middle point. The training set is split, depending upon which side of hyperplane the points lie. Kuncheva and Rodriguez [10,12] suggest two reasons for the success of RLO ensembles.

1. As linear oracle splits the space into two subspaces the classification task is easier, which may lead to the better classification accuracy (for pair of classifiers) than the classifier trained on complete space.
2. The second reason is that with random linear oracles diverse classifiers are created.

The performance of RLO ensembles with Naive Bayes classifiers is better than other popular ensemble methods [12]. Whereas, use of decision trees, in the RLO

```
Random Linear Oracle Algorithm
Initialisation- Choose the ensemble size M, the base classifier model D and classification problem P defined as a labelled training set T and a set of classes $\Omega$.

Ensemble Construction
for i=1...M do
    a- Draw a random hyperplane $h_t$ in the feature space of P.
    b- Split the training set T into $T^+$ and $T^-$ depending on which side of $h_t$ the points lie.
    c- Train a classifier for each side, $D_i^+ = D(T^+, \Omega)$ and $D_i^- = D(T^-, \Omega)$.
    Add the mini-ensemble of the two classifiers and the oracle, $(h_t, D_i^+, D_i^-)$, to the current ensemble.
end for
Classification
a- For a new object $\mathbf{x}$, find the decision of each ensemble member by choosing $D_i^+$ or $D_i^-$ depending on which side of $h_t$, $\mathbf{x}$ is.
b- Combine the decisions of all the selected classifiers by the chosen combination rule (we use majority voting scheme).
```

Fig. 2. Random Linear Oracle (RLO) Algorithm

ensembles [10], does not give this much improvement. That leads to a question, why RLO ensembles with Naive Bayes classifiers perform so well. The answer lies in the property of Naive Bayes classifier. As we have discussed in the last section that Naive Bayes classifier is a global classifier. If the classes of a datasets spreads in multiple regions, the estimation of class-conditional probability is biased and Naive Bayes performs poorly [15]. Decomposing each class into many clusters may be beneficial in these cases [15].

A random linear oracle divides the training dataset into two clusters, each in different regions. Hence, there is a large probability that those classes that spread in multiple regions will be divided into two subclasses, each belongs to the one of the clusters created by the random linear oracle. This introduces a form of class locality in Naive Bayes classifier. We get better class probability estimates in this case. That is the probable reason for the good performance of RLO ensembles with Naive Bayes classifiers. Decision trees are capable of representing complex decision boundaries. Hence, the subclasses strategy has not been very useful for decision trees. RLO ensembles with decision trees do not produce good performance [10] that is because individual decision trees in a RLO ensemble are not benefited much from the division of the data.

Creating subclasses in the data is helpful in improving the performance of low variance classifiers (eg. linear classifiers) [14]. This is useful for datasets that consist of classes having more than one intrinsic component.

That suggests that RLO ensembles are more useful for linear classifiers. In the next section, we present proposed Random Subclasses (RS) method that divides each class into a set of subclasses using the RLO framework to show that it is the creation of subclasses that is helpful in good performance of RLO ensembles with Naive Bayes classifiers.

4 Random Subclasses (RS)

In this section, we propose a new ensemble method Random Subclasses (RS) which is quite similar to RLO, however RS is more computationally efficient and flexible as compared to RLO. The purpose of this algorithm is not to present an algorithm that is better (in terms of classification accuracy) than RLO but to emphasise the fact that the creation of subclasses is helpful in RLO.

We suggest RS method that creates subclasses directly. To create 2 subclasses from a class, we select 2 points randomly from the data points belonging to the class and use RLO framework to create 2 subclasses from the data points belonging to the class. We create subclasses for all classes available in the dataset (Fig. 3). When K hyperplanes are same in a RS classifier, it will produce the same structures as in a RLO classifier (Fig. 4). After creating subclasses, we train a classifier on the complete data treating the subclasses as the classes of the data (for K classes 2K subclasses are created). In RS method, the prediction by each classifier is produced in new output space (new subclasses). These predictions are converted to original class labels as the relationship between the classes and the subclasses is known. The RS algorithm is shown in Fig. 5.

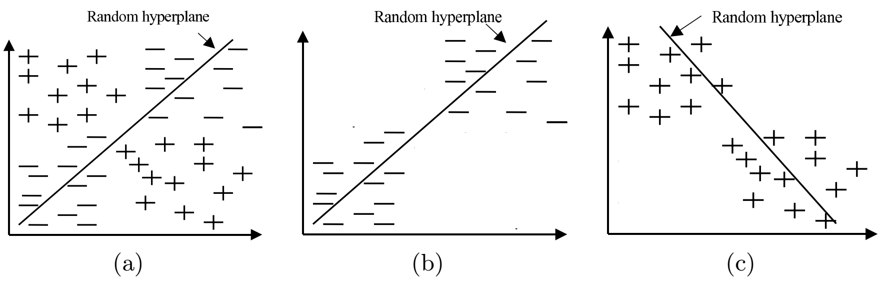


Fig. 3. Two class problem (a) The RLO framework divides the complete data into two subspaces using a random hypeplane. (b) and (c) show RS in which we create a random hyperplane for each class and each class is divided into two subclasses.

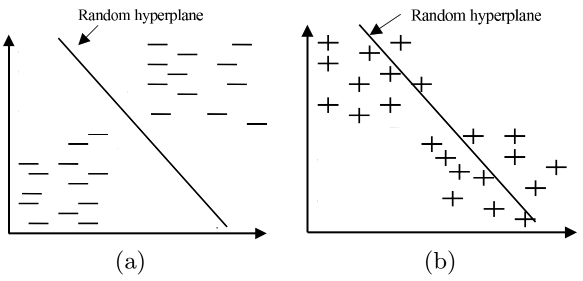


Fig. 4. When hyperplanes for both the classes are same, RS creates the same structures as RLO

The main difference between RLO and RS is that in RS we create K hyperplanes (K is the number of classes), one for each class whereas in RLO method we have one hyperplane. RLO divides the entire training data into two subspaces whereas RS divides each class into two subclasses. A RLO classifier has a pair of classifiers, these two classifiers learn on different subspaces decided by a hyperplane, whereas a RS classifier is a single classifier trained on the complete data. During testing in the RLO method, first the position of the testing data point is decided and the decision of the classifier, which is trained in that subspace, is used. In RS, we do not have this selection step as we have only one classifier. The RS approach has following advantages over the RLO approach;

1- In RLO, for an ensemble of size M, 2M classifiers are trained, whereas in RS only M classifiers are trained.
2- We have better flexibility in RS for creating subclasses which is the reason of the improved performance of RLO ensembles. In RS, we can control the number of subclasses per class (2 subclasses or no subclasses), whereas we do not have this kind of control in RLO. For example, if the number of data points is low in a class, we may not create subclasses for that class.

```
Algorithm -Random Subclasses
Input: Choose the ensemble size M, training data T with K classes, a testing data point x.
Output: Prediction of x.
Ensemble Construction-
for i=1...M do
  a- Separate T into subsets $T_j$ where $T_j = \{(x, y) \in T \mid y = y_j\}$
  for j=1...K do
    a- Draw a random hyperplane $h_j$ (using the method suggested in RLO method) in the feature space of $T_j$.
    b- Create two new subclasses depending on which side of $h_j$ the points lie for $T_j$ points.
  end for
  After this step, we have 2K classes (a new output space).
  Train classifier on this dataset.
end for
Classification
a- Find the decision of each ensemble member.
b- Convert it into original class label.
c- Combine the decisions of all the selected classifiers by the chosen combination rule (we use majority voting scheme).
```

Fig. 5. Random Subclasses Method

3- Take the case of prediction of a data point, which has some of the attribute values missing. Naive Bayes can work with missing attribute values. In RLO method, the position of testing data point is computed, this is an additional source of error as an inaccurate selection of the classifier may lead to the poor classification result.

One may argue that the method (random hyperplanes) used in the proposed RS method to create subclasses is not very accurate and in some cases may give very bad subclasses that may not be useful. In these cases, we may not be getting improvement in the classifier performance. We are creating ensembles of classifiers and the final result in an ensemble is the combination of many classifiers. Hence, even if some of the individual classifiers may not have improved performance, the ensembles perform well because of the classifiers that have improved performance.

RLO and RS are similar in the sense that both create new class boundaries by using random hyperplanes. In the next section, we present experimental results.

5 Experiments

We investigate two hypotheses in these experiments. One, whether RS is similar to RLO for Naive Bayes classifier as both create subclasses using the same method (random hyperplanes) and the second that RS and RLO are useful for the other low variance classifiers. For the second hypothesis, we used SVM with linear kernel classifier for our experiments.

Table 1. Datasets used in experiments

Dataset Name	Size	No. of Classes	No. of cont. attributes	No. of Cat. attributes
DNA	3175	3	-	60
Glass	215	7	9	-
Ionosphere	351	2	34	-
Optical	5620	10	64	-
Pendigits	10992	10	16	-
Pima-diabetes	768	2	8	-
Ring-Norm	7400	2	20	-
Satimage	6435	6	36	-
Segment	2310	7	19	-
Sonar	208	2	60	-
Spam	4601	2	57	-
Tic-Tac-Toe	958	2	-	9
Two-Norm	7400	2	20	-
Vehicle	846	4	18	-
Vote	435	2	-	16
Vowel	990	11	10	-
Waveform	5000	3	40	-

We carried out experiments on 17 datasets taken from UCI Repository. Information about datasets are shown in Table 1. We carried out our experiments using Naive Bayes classifier and SVM with linear kernel as the base classifier. We carried out experiments with Bagging [2] and AdaBoost.M1 [7] modules of WEKA [16]. The size of the ensemble was fixed to 25 as mentioned by Kuncheva and Rodriguez [12]. For a RLO ensemble, 25 pairs of classifiers were trained.

We performed five replications of a two-fold cross-validation. In each replication, the dataset was divided into two random equal-sized sets. Each learning algorithm was trained on one set at a time and its error was estimated on the other set. The original test proposed by Dietterich [3] to compare the performance of classifiers suffers from low replicability. Alpaydin [1] proposes a modification to the 5×2 cross-validation F test. That test was used in our experiments. A confidence level of 95% for this test was considered.

To calculate the hyperplane and for determining on which side of it a given point lies, each categorical attribute of was replaced by $|A|$ binary attribute where $|A|$ is the number of possible categories and all numeric attributes were linearly scaled within the interval [0,1] [10].

We used WEKA [16] implementation of Naive Bayes classifier for our experiments. For the numeric data, normal distribution option was selected. Results are presented in Table 2. Results show that RS is similar to RLO (3 Wins, 14 Draws) this verifies our hypothesis that RS is creating the same effect as RLO. RS performed statistically better than or similar to Bagging (10 Wins and 7 Draws), AdaBoost.M1 (9 Wins, 5 Draws and 3 Losses) and a single Naive Bayes classifier (10 Wins and 7 Draws) for most of the datasets.

Table 2. Classification error in % for different ensemble methods with Naive Bayes classifier as the base classifier. Number in bold shows the best performance for that dataset. "+" shows that performance of RS is statistically better than that ensemble method for the dataset. "-" shows that performance of RS is statistically worse than that ensemble method for the dataset.

Data	RS	RLO	Bagging	AdaBoost.M1	Single classifier
DNA	**4.6**	**4.6**	4.8	8.1(+)	4.8
Pima Dia	**22.9**	23.2	23.6	24.2	23.5
Glass	**37.6**	38.2	48.3(+)	50.3(+)	51.6(+)
Ionosphere	17.7	16.3	21.1(+)	**9.5**(-)	20.7(+)
Optical	**6.8**	**6.8**	8.8(+)	8.6(+)	8.8(+)
PenDigit	8.1	**8.0**	14.1(+)	14.2(+)	14.2(+)
Ring-Norm	2.3	**2.2**	**2.2**	2.3	**2.2**
Satimage	**15.1**	17.1(+)	20.2(+)	20.4(+)	20.5(+)
Segment	**17.2**	17.8	19.6	18.2	19.8
Sonar	24.4	**23.5**	28.0	20.8	29.1
Spambase	**18.7**	19.6(+)	20.0(+)	19.9(+)	20.0(+)
Tic-Tac-Toe	26.7	27.0	28.1	**8.1**(-)	28.4
Two-Norm	**2.2**	**2.2**	**2.2**	**2.2**	**2.2**
Vehicle	35.8	**34.1**	53.9(+)	54.5(+)	54.5(+)
Vote	7.7	7.3	10.6(+)	**4.7**(-)	10.7(+)
Vowel	**21.7**	25.1(+)	35.0(+)	27.9(+)	35.9(+)
Waveform	**14.8**	16.0	20.1(+)	19.4(+)	20.1(+)
Win/Draw/Loss		3/14/0	10/7/0	9/5/3	10/7/0

We used WEKA [16] implementation of Support Vector Machine with linear kernel with regularization parameter C = 1 (default value) for our experiments. Results are presented in Table 3. Table 3 shows that for 7 out of 17 datasets, RS and RLO performed statistically better than a single classifier. Bagging improved the performance for DNA data whereas AdaBoost.M1 did not improve the performance for any of the datasets. Creating subclasses is the reason for the success of RLO and RS. As new subclass boundaries create easier to learn problems, the subclasses strategy works well with linear classifiers.

Experiments results verify both hypotheses that RS and RLO are similar and RS and RLO are useful for the other low variance classifiers. The similar performance of RLO and RS shows that the creation of new decision boundaries is the main reason of the success of RLO with low variance classifiers.

The better performance of RLO can be explained by using bias-variance properties of RLO classifiers. The creation of subclasses increases the number of decision boundaries per class, this increases the complexity of individual classifiers, hence it reduces the bias part of the classifier error [14]. It also increases the variance part of the error. RLO classifiers are diverse classifiers due to different random hyperplanes, an ensemble of these diverse classifiers reduces the variance part of the error [2]. The same augment can be used to explain the performance of RS.

Table 3. Classification error in % for different ensemble methods with SVM with linear kernel as the base classifier. Number in bold shows the best performance for that dataset. "+" shows that performance of RS is statistically better than that ensemble method for the dataset. "-" shows that performance of RS is statistically worse than that ensemble method for the dataset.

Data	RS	RLO	Bagging	AdaBoost.M1	Single classifier
DNA	**5.1**	**4.9**	6.0(+)	7.5(+)	7.8(+)
Pima Dia	23.1	**22.3**	22.8	22.8	22.9
Glass	48.0	45.7	48.1	**44.4**	49.2
Ionosphere	11.9	12.3	12.3	**11.3**	12.5
Optical	**1.5**	1.6	1.8(+)	2.3(+)	2.1(+)
PenDigit	**1.3**	1.5(+)	2.4(+)	2.2(+)	2.5(+)
Ringnorm	**20.9**	**20.9**	22.7(+)	22.3(+)	22.8(+)
Satimage	**12.8**	13.3(+)	13.7(+)	13.7(+)	13.7(+)
Segment	7.8	**7.4**	7.9	7.7	7.7
Sonar	19.0	**18.7**	22.9(+)	23.1(+)	24.7(+)
Spambase	10.3	**9.7**(-)	9.8(-)	9.9	10.1
Tic-Tac-Toe	**1.6**	**1.6**	**1.6**	2.4(+)	**1.6**
Two-Norm	**2.2**	**2.2**	**2.2**	**2.2**	**2.2**
Vehicle	28.1	**27.4**	28.7	29.0	29.0
Vote	3.9	4.0	**3.8**	5.0(+)	4.1
Vowel	31.6	**29.9**	39.2(+)	40.2(+)	40.3(+)
Waveform	14.1	14.0	**13.9**	14.1	14.1
Win/Draw/Loss		2/14/1	7/9/1	9/8/0	7/10/0

6 Conclusion and Future Work

Most of the ensemble methods require unstable base classifier. Naive Bayes is a stable classifier. Hence, it is not suited for as a base classifier for ensembles. However, RLO performs well with Naive Bayes classifiers. In this paper, we investigate the reason of RLO success. We conclude that the creation of local structures in a RLO classifier leads to the better class probabilities estimates hence improves the performance for Naive Bayes Ensembles. We also develop RS ensemble method that decomposes classes into subclasses. RS is less computationally expensive and more flexible as compared to RLO. We carried out a comparative study on 17 datasets taken from UCI repository. Results suggest that RS and RLO are similar, whereas RS and RLO are statistically better than or similar to Bagging and AdaBoost.M1 for most of the datasets.

We also carried out the comparative study of different ensemble methods using SVM with linear kernel as the base classifier. RLO and RS showed improvement for some datasets whereas for almost all datasets Bagging and AdaBoost.M1 did not show any improvement over a single classifier. We conclude from our study that RLO and RS are useful for classifiers that can not learn complex class boundaries e.g. linear classifiers.

RLO has been employed with other ensemble methods [10,12]. Results suggest that all ensemble methods in study, with Naive Bayes as the classifiers, are better with RLO than their standard versions. Combining RS with other ensemble methods will be one of the future directions of this study.

References

1. Alpaydin, E.: Combined 5 x 2 cv f Test Comparing Supervised Classification Learning Algorithms. Neural Computation 11(8), 1885–1892 (1999)
2. Breiman, L.: Bagging Predictors. Machine Learning 24(2), 123–140 (1996)
3. Dietterich, T.G.: Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation 10, 1895–1923 (1998)
4. Domingos, P., Pazzani, M.: On the Optimality of the Simple Bayesian Classifier under Zero-one Loss. Machine Learning 29, 103–130 (1997)
5. Eick, C.F., Nidal, Z.: Using Supervised Clustering to Enhance Classifiers. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS, vol. 3488, pp. 248–256. Springer, Heidelberg (2005)
6. Freund, Y.: Boosting a Weak Learning Algorithm By Majority. Information and Computation 121(2), 256–285 (1995)
7. Freund, Y., Schapire, R.E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
8. Hand, D.J., Yu, K.: Idiot's Bayes - Not so Stupid After All. International Statistical Review 69, 385–399 (2001)
9. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
10. Kuncheva, L.I., Rodriguez, J.J.: Classifier ensembles with a random linear oracle. IEEE Trans. on Knowledge and Data Engineering 19(4), 500–508 (2007)
11. Rish, I., Heellertein, J., Jayram, T.: An Analysis of Naive Bayes on Low-Entropy Distributions, Tech. Report RC91994, IBM T. J. Watson Research Center (2001)
12. Rodriguez, J.J., Kuncheva, L.I.: Naive bayes ensembles with a random oracle. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 450–458. Springer, Heidelberg (2007)
13. Tumer, K., Ghosh, J.: Error Correlation and Error Reduction in Ensemble Classifiers. Connect. Sci. 8(3), 385–404 (1996)
14. Vilalta, R., Achari, M.R., Eick, C.F.: Class Decomposition via Clustering: A New Framework for Low Variance Classifiers. In: ICDM 2003 (2003)
15. Vilalta, R., Rish, I.: A Decomposition of Classes via Clustering to Explain and Improve Naive Bayes. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS, vol. 2837, pp. 444–455. Springer, Heidelberg (2003)
16. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques., 2nd edn. Morgan Kaufmann, San Francisco (2005)

Stacking for Ensembles of Local Experts in Metabonomic Applications

Kai Lienemann, Thomas Plötz, and Gernot A. Fink

TU Dortmund University, Intelligent Systems Group, Germany
{Kai.Lienemann,Thomas.Ploetz,Gernot.Fink}@udo.edu

Abstract. Recently, Ensembles of local experts have successfully been applied for the automatic detection of drug-induced organ toxicities based on spectroscopic data. For suitable Ensemble composition an expert selection optimization procedure is required that identifies the most relevant classifiers to be integrated. However, it has been observed that Ensemble optimization tends to overfit on the training data. To tackle this problem we propose to integrate a stacked classifier optimized via cross-validation that is based on the outputs of local experts. In order to achieve probabilistic outputs of Support Vector Machines used as local experts we apply a sigmoidal fitting approach. The results of an experimental evaluation on a challenging data set from safety pharmacology demonstrate the improved generalizability of the proposed approach.

1 Introduction

In the last two decades the development of new NMR (nuclear magnetic resonance) measurement techniques together with a steadily increasing spectral resolution and improved data quality, respectively, have provided the opportunity for in-depth automatic analysis of biofluids. Thereby, the ultimate goal is to detect specific changes of an organism's metabolism that is, for example, induced by drug applications in safety pharmacology. Generally, the research field of Metabonomics addresses "the quantitative measurement of the time-related multiparametric metabolic response of living systems to pathophysical stimuli or genetic modification" [1]. In addition to classical analysis methods from clinical chemistry and histopathology meanwhile also automatic classification techniques utilizing pattern recognition approaches have been applied successfully.

Recently, multiple classifier systems have been developed for the detection of drug-induced organ toxicities with applications to industrial safety pharmacology (cf. e.g. [2]). It has been shown that the use of Ensemble methods that integrate multiple classifiers each providing local views on the spectra outperforms single classifier approaches. However, when comparing the classification performance of classifier Ensembles achieved on cross-validation data with those on test-sets it becomes clear that the systems tend to overfit. This is especially critical when only small portions of NMR spectra are available for training and optimization.

In this paper we present an enhancement of Ensembles of local experts for Metabonomic applications that explicitly focuses on improved generalizability.

Our goal is to stabilize the classification performance from cross-validation to test. Therefore, a variant of stacked generalization [3] as a combination method of predictions from different models is integrated into our Ensemble system for NMR classification. Thereby, the decisions of local experts that focus on small parts of the spectra serve as probabilistic level-0 model outputs. Since for this level of classification we use Support Vector Machines that generate binary decisions pseudo-probabilities are derived by means of a sigmoidal mapping approach. Subsequently, an additional classifier – the level-1 generalizer – is applied to the vectors of pseudo-probabilities providing the final classification result. In order to find the most suitable configuration we investigated the appropriateness of certain variants of level-1 generalizers. By means of an experimental evaluation on a realistic NMR dataset from industrial safety pharmacology we demonstrate the effectiveness of the proposed approach. Using stacking for Ensembles of local experts improved generalization can be achieved for Metabonomic applications.

In the following section the general background for the automatic analysis of NMR spectra is given together with the motivation of our current work. Subsequently in section 3 the proposed Ensemble system that focuses on improved generalizability for toxicity prediction is described. The results of the experimental evaluation are presented in section 4. The paper ends with a conclusion.

2 Background and Motivation

Within an NMR spectrum of some analyzed sample the concentration of numerous molecules is represented by peak intensities. Peak positions are specific for the respective molecules as exemplary shown in figure 1. Changes in the concentration of several molecules can be detected by comparison of corresponding peak intensities between different samples. Substantial changes indicate an alteration of the organism's metabolic profile. Consequently, a major issue in Metabonomics research is the development of systems for an automatic analysis of samples for the identification of relevant peak changes.

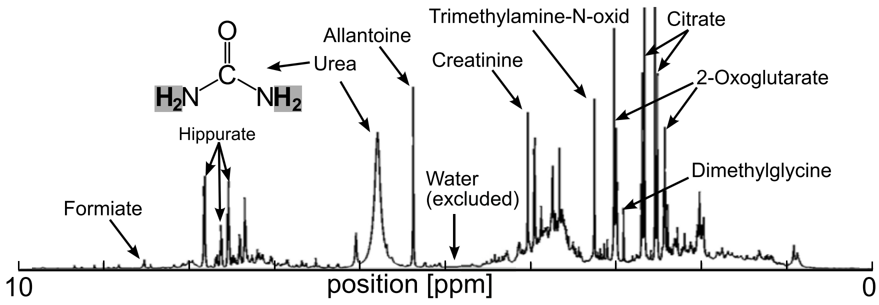


Fig. 1. Exemplary ^{1}H-NMR spectrum of an urine sample from an untreated rat. A subset of peaks and their corresponding molecules, the chemical structure of urea and the signal emitting hydrogen atoms are denoted.

Recently pattern recognition techniques have also been applied to Metabonomics tasks. It has been demonstrated that, generally, it is possible to detect changes in metabolisms by comparison of spectroscopic data (cf. [4,5,6]). Certainly the most promising approach – *Classification of Unknowns by Density Superposition* [7] – was developed within the *Consortium for Metabonomic Toxicity (COMET)* project [8]. This approach is based on spectra classification using probabilistic neural networks [9] that were trained exploiting a large database which is not publicly available. When generally analyzing related work as it has been reported in the literature it becomes clear that so far only little research has been devoted to the automatic classification of NMR data.

In our previous work for the first time Ensemble methods were developed for the prediction of organ toxicities based on spectroscopic data [10]. We developed a multiple classifier system that introduced weighted Random Subspace Sampling (RSS) with applications to the field of Metabonomics. The weights of variables relevant for toxicity classification were iteratively optimized by an unsupervised learning approach. Subspaces were classified by Support Vector Machines (SVMs) and aggregation of the predictions was performed by majority voting. It has clearly been shown that differences in the relevance of distinct variables, i.e. parts of the NMR spectra, exist w.r.t. their significance for classification. Favoring relevant regions in RSS finally improved the Ensemble classification performance. An alternative multiple classifier approach for the analysis of NMR data has been proposed in [2] where preprocessing methods were varied for Ensemble creation. Again it has been shown that the combination of multiple classifiers outperforms single classifier approaches in Metabonomics. The idea of focusing on certain spectral regions for classification and their combination in an Ensemble system has been further investigated in [11]. In this approach an Ensemble of local experts is created by training classifiers on short spectral regions that are determined by a sliding window technique. Final aggregation of local experts' predictions is achieved by majority voting, whereas the subset of experts used for final voting is optimized in order to achieve an improved classification accuracy. Thus, the classification decision is based on specific parts of the spectra, which are supposed to reflect biologically relevant changes.

Ensemble optimization for automatic analysis of NMR data has achieved a nearly perfect classification performance on the validation sets. Unfortunately, the generalization capabilities of the system on this extremely challenging type of data is still not optimal. An experimental evaluation of several strategies for ensemble optimization has indicated a generally decreasing classification performance on unknown test data [11]. A promising approach for the combination of multiple models emphasizing the idea of cross-validation for an improvement of generalization capabilities – *generalized stacking* – was proposed by Wolpert [3], and further discussed by Breiman [12] and Ting et al. [13] (cf figure 2). In this approach a given data set is split into J training and test sets according to the J-fold cross-validation principle. Different so-called level-0 models are estimated on the training sets and applied to predict the corresponding test sets. The predictions from all test sets are collected in a new data set and further used as

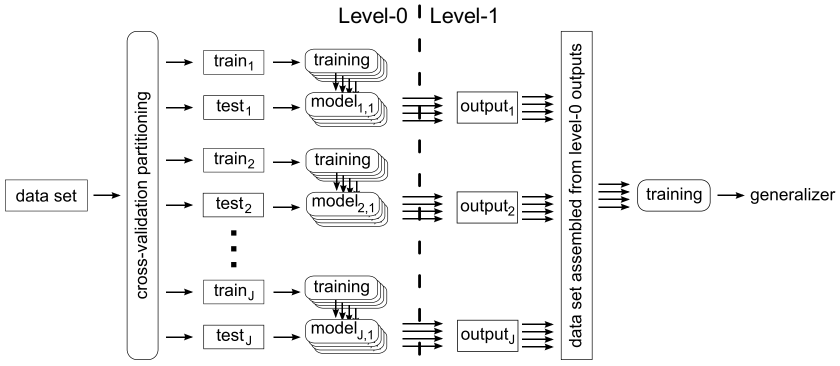


Fig. 2. Illustration of the generalized stacking approach. A data-set is subdivided into J cross-validation parts and 4 models are trained on each set, respectively. The outputs of different sets are collected in a single data set for training the final classifier.

input for training of a final classifier, the so-called level-1 generalizer. For a final classification level-0 models are trained on the whole data set and new samples are classified by the level-1 generalizer based on the predictions of level-0 models. Generally, the type of level-0 model and level-1 generalizer is not restricted to any specific classification algorithm. However, Ting et al. assume probabilistic outputs of level-0 models rather than class predictions for a successful application of generalized stacking [13]. By means of generalized stacking the severe problem of overfitting in Ensemble optimization methods can be avoided, which is a clear advantage contrary to optimization of expert selection for majority voting as a nontrainable combination method.

3 Stacking for Fusion of Local Spectral Information

In order to stabilize the classification results of the multiple classifier system of local experts for the analysis of NMR spectra we developed an Ensemble estimation approach that explicitly focuses on improved generalizability. Therefore, the problem of overfitting is tackled by the application of a trainable combiner which is inspired by the concept of generalized stacking. Consequently, the new approach is referred to as *stacking for Ensembles of local experts*. The basic idea is to use the output of local classifiers that represent statistical models for designated parts of NMR spectra as input data for a second classifier. The parametrization of the level-1 generalizer is adjusted by cross-validation. Thereby, rules for classification according to the local experts' predictions are automatically derived by the stacked classification algorithm. Figure 3 gives an overview of the system.

The new approach of stacking for local experts in Metabonomic applications extends our previous work on Ensembles of local experts for the automatic analysis of NMR data [11]. Local information of spectroscopic data is treated by

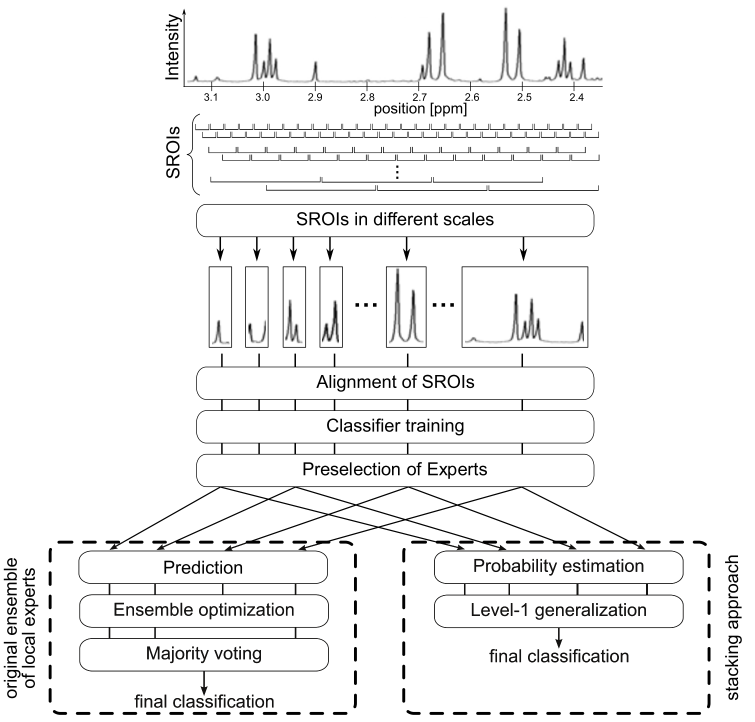


Fig. 3. Classification system for automatic analysis of NMR data based on an Ensemble of local experts and stacking for improved generalizability (see text for description)

classifiers that have restricted views on the data thereby covering very few peaks only. Spectral regions of interest (SROIs) are determined on NMR data by applying a sliding window approach and an alignment procedure, respectively, in order to compensate peak shifts induced by changes of physiochemical factors like pH or ion concentration. SVMs using a radial basis function kernel are trained for each SROI of spectral intensities and serve as local experts for specific regions. Initially experts exhibiting insufficient classification performance on a validation set are excluded from the Ensemble and selection of experts for Ensemble aggregation by majority voting is optimized in a final step in order to achieve a suitable classification accuracy.

Only very few substances (or combinations of substances) that indicate induced organ toxicities after drug application (by changes in their concentration) are known in safety pharmacology. This fact supports the assumption that even though concentration information of numerous molecules is present in an NMR spectrum only a very small fraction is useful for the detection of drug-induced organ toxicities. Thus, the identification of a suitable combination of local experts for the final classification is the most crucial step in the Ensemble of local experts. Unfortunately, this process is prone to overfitting.

Dos Santos et al. proposed to use a genetic algorithm approach for (general) Ensemble optimization which reduces the tendency to overfitting [14]. They found that a cross-validation procedure has to be used in order to evaluate the classification performance on an independent validation set. However, analyzing our task it becomes clear that cross-validation is not generally applicable to majority voting. Thus, we focus on an alternative aggregation approach aiming at improved generalizability while retaining high classification performance.

According to the terminology used in the stacked generalization literature in our approach SVMs, serving as local experts on NMR data, are used as level-0 models. According to [13] for best generalization the level-1 generalizer should have probabilistic input, i.e. level-0 classification have to provide probabilities rather than binary decisions. The latter is, however, the case for standard Support Vector Machines as they generate -1 or 1 decisions, respectively, depending on the position of the test sample w.r.t. the hyperplane that separates the particular classes. In order to achieve probabilistic SVM decisions we integrate an additional post-processing step into SVM classification. As developed by Platt [15], and further refined by Lin and coworkers [16] probabilities can be generated from SVM decisions by fitting a parametrized sigmoidal function to the distances of the samples from the labeled training set to the separating hyperplane. The rationale is based on the assumption that greater distances indicate higher confidences for the classification. Analogously smaller distances correspond to smaller confidences. In our approach SVM probabilities derived in this way represent the output of the local experts which is fed into the level-1 classifier.

To sum up, the original Ensemble of local experts approach is extended using a stacked classifier on the outputs of the local experts for final classification. Furthermore, probability estimates of SVMs are integrated in the Ensemble system and serve as input for the level-1 generalizer. These enhancements are supposed to increase the generalization of the Ensemble of local experts, which will be shown by an experimental evaluation and comparison to previous results.

4 Experimental Evaluation

In order to evaluate the effectiveness of the new approach with respect to the addressed improved generalizability we performed various practical experiments. As in our previous work they are related to the detection of drug-induced organ toxicities based on a challenging real-world data set from pharmaceutical industry[1]. This set contains 896 ^{1}H NMR spectra of urine samples from rats treated with one of 53 pharmaceuticals. According to literature investigations and histological judgments induced organ toxicity regarding proximal tubulus (kidney) is present in 259 samples (= 18 pharmaceuticals). Details on spectra measurements, data treatment and histological judgment are given in [11].

[1] The presented evaluation is restricted to this data set due to the lack of publicly available data sets of NMR spectra. The presented approach is, however, not dependent on the type of data used and generally applicable.

The presentation of the results is structured as follows. First we determine the optimal configuration of the proposed approach by evaluating the suitability of various level-1 generalizers, measuring the impact of probabilistic level-0 model outputs for our task. Subsequently the classification capabilities are directly compared with those that have been achieved using the original Ensemble of local experts thereby clearly indicating the improved generalizability of the proposed method for the particular test sets. For all experiments the selection of SROIs, training of local experts and their preselection for Ensemble generation, resulting in 147 local experts for the final Ensemble, were performed as previously [11].

We performed a five-fold cross-validation and test procedure for training, parameter optimization and final test. Samples were grouped according to target and indication of their corresponding pharmaceutical. These groups of samples are sub-divided into five sets while trying to keep ratios of non-toxic and toxic samples approximately equal. Training was pursued using three fifths of the sets, parameter optimization by one fifth and final testing on the remaining set in every possible configuration. Final classification rates are given as averages over the results on the particular sets. In addition to focusing on specific samples a further goal is the classification of *pharmaceuticals* as being toxic or non-toxic. Thus, results for analyzed samples that have been collected at different time-points are aggregated to final classifications of the corresponding pharmaceutical (maximum mean value) – referred to as group-classification. Due to its robustness to imbalanced data-sets the *Matthews Correlation Coefficient* [17] – MC – (normalized to $[-1 \dots 1]$) has been used as primary evaluation criterion for all training and optimization procedures. For completeness also classification accuracy (acc), specificity (spec) and sensitivity (sens) values are shown.

Table 1 summarizes the classification results achieved for different level-1 generalizers based on probabilistic level-0 model outputs. Results for cross-validation and test using either a k nearest neighbor classifier (kNN – with $k = 7$ optimized according to a fixed grid from one to 31), grid-search optimized SVMs with linear (LSVMs) or radial basis kernel functions (RSVMs), respectively, and random forests (RFs) [18] are shown. RFs are parametrized depending on the data dimensionality v, using $\lceil log_2(v) \rceil$ decision trees in the forest and selecting $\lceil \sqrt{v} \rceil$ variables randomly at each node. Analyzing the results (level of significance for all sample-based experiments: $\approx \pm 2.5\%$) it can be seen that RSVMs outperform all other techniques. Furthermore, it becomes clear that improved stability of classification results when turning from cross-validation towards test is gained independently of the particular choice of level-1 generalizer.

In order to validate the necessity of probabilistic outputs of level-0 models for a generalizing stacking system (as claimed by Ting et al. [13] – cf. section 3) the new approach was also evaluated using binary predictions of local experts. By means of the results presented in table 2 the assumption of Ting et al. can clearly be confirmed also for the Metabonomic application case. Using Ensembles with probabilistic level-0 outputs better classification accuracies together with improved generalizability can be achieved.

Table 1. Classification performance of different level-1 generalizer algorithms based on probabilistic outputs of RSVMs as level-0 models in the Ensemble of local experts

	cross-validation				test			
Measure	***k*NN**	**RF**	**LSVM**	**RSVM**	***k*NN**	**RF**	**LSVM**	**RSVM**
acc [%]	82.7	77.0	82.4	**83.5**	80.7	76.7	80.7	**81.0**
spec [%]	**95.4**	92.5	94.5	94.4	**93.9**	92.8	93.3	92.0
sens [%]	51.4	39.0	52.5	**56.8**	48.3	36.7	49.8	**54.1**
MC	0.551	0.383	0.542	**0.575**	0.494	0.366	0.496	**0.510**

Table 2. Classification performance using probabilistic or prediction outputs of RSVMs as local experts and RSVM for stacked classification

	cross-validation		test	
Measure	**probabilistic**	**prediction**	**probabilistic**	**prediction**
acc [%]	**83.5**	76.9	**81.0**	61.7
spec [%]	**94.4**	91.1	**92.0**	73.8
sens [%]	**56.8**	42.1	**54.1**	32.1
MC	**0.575**	0.387	**0.510**	0.058

Table 3. Changes in classification performance using a PLS transformation prior to classification by RSVM

	cross-validation		test	
Measure	**RSVM**	**PLS + RSVM**	**RSVM**	**PLS + RSVM**
acc [%]	**83.5**	**83.5**	81.0	**82.6**
spec [%]	**94.4**	90.0	**92.0**	88.2
sens [%]	56.8	**67.6**	54.1	**68.7**
MC	0.575	**0.590**	0.510	**0.574**

Although the preselection of local experts already reduces the set of experts used for stacked classification to those with a reasonable classification accuracy, a further selection of experts is beneficial as shown in the original Ensemble of local experts approach. A well-known method for variable weighting of a labeled multidimensional data set according to the relevance of the variables for class separation is the projection to latent structures (PLS, also referred to as partial least squares) [19]. PLS transformation is comparable to principal component analysis (PCA) differing, however, in the optimization criterion. For PLS it is not the explained variance of the new coordinate system that is optimized but the covariance between the data variables and class labels. Thus, PLS focuses on variables (in this case local experts) relevant for class discrimination, thereby achieving an implicit weighting of experts. The application of the PLS transformation on the probabilistic outputs prior to classification by a RSVM leads to an improved classification accuracy on the cross-validation set and only a slight decrease on the test set can be observed (cf. table 3).

Table 4. Evaluation of the original Ensemble of local experts approach and the proposed stacking modification on cross-validation (xval) and test. Bold numbers indicate the best performance of the two methods on the respective set, while italic numbers highlight the best relative change (Δ) when comparing cross-validation and test.

	Local Experts			**Local Experts + Stacking**		
Measure	xval	test	Δ	xval	test	Δ
			sample classification			
acc [%]	**86.3**	77.8	-9.9 %	83.5	**82.6**	*-1.1 %*
spec [%]	**99.2**	**94.2**	-5.0 %	90.0	88.2	*-2.0 %*
sens [%]	54.4	37.5	-31.1 %	**67.6**	**68.7**	*+1.6 %*
MC	**0.659**	0.402	-39.0 %	0.590	**0.574**	*-2.7 %*
			group classification			
acc [%]	**98.1**	88.5	-9.8 %	92.3	**90.4**	*-2.1 %*
spec [%]	**100**	94.1	-5.9 %	**100**	**97.1**	*-2.9 %*
sens [%]	**94.4**	**83.3**	-11.8 %	77.8	77.8	*±0 %*
MC	**0.958**	0.785	-18.1 %	0.834	**0.786**	*-6.1 %*

The final part of the evaluation addressed a direct comparison of the results achieved using either the original Ensemble of local experts approach or the enhanced version integrating the proposed stacking technique. The results presented in table 4 clearly indicate the improved generalization of the latter. The classification accuracies on cross-validation slightly decrease or remain almost the same (level of significance for group classification: $\approx \pm 8.2\%$) but the effect of overfitting as it was observed for the original Ensemble of local experts can almost be eliminated when using the new approach. This improved generalizability is of major importance for safety pharmacology where unknown samples need to be classified reliably.

5 Conclusion

Generalizability of an automatic classification system is a major prerequisite for its application to real-word problems. In this paper we presented a new model combination approach that does not suffer from the problem of overfitting during optimization as observed for our previously developed Ensemble of local experts approach for Metabonomic applications. Motivated by the concept of generalized stacking outputs of local NMR experts are aggregated for final Ensemble estimation. Improved generalizability is achieved by parameter optimization using a cross-validation approach in a hierarchical classification framework. The decisions of local classifiers – level-0 models – serve as input for an additionally subsequent level-1 generalizer. Generally, stacked generalization works best when integrating level-0 models that generate probabilistic outputs. Consequently, in our approach decisions of Support Vector Machines are transformed from binary towards pseudo-probabilistic by means of a sigmoidal fitting function.

The effectiveness of stacking for Ensembles of local experts for Metabonomic applications was demonstrated in an experimental evaluation. Analyzing a challenging real-world set of NMR spectra from industrial drug design shows improved generalizability of the classification performance when turning from cross-validation towards test. In the typical use-case of an automatic classification system in safety pharmacology toxicities need to be predicted reliably for unknown samples. Thus, reducing the effect of overfitting is of major importance which clearly emphasizes the practical relevance of the proposed approach.

Acknowledgments. Parts of this work have been funded by a grant from Boehringer Ingelheim Pharma GmbH & Co. KG. (BI), Genomics group. We would like to thank the General Pharmacology Group of BI for providing the data.

References

1. Nicholson, J.K., et al.: Metabonomics: a platform for studying drug toxicity and gene function. Nature Reviews Drug Discovery 1, 153–161 (2002)
2. Lienemann, K., Plötz, T., Pestel, S.: NMR-based urine analysis in rats: Prediction of proximal tubule kidney toxicity and phospholipidosis. Journal of Pharmacological and Toxicological Methods 58, 41–49 (2008)
3. Wolpert, D.H.: Stacked generalization. Neural Networks 5, 241–259 (1992)
4. Holmes, E., et al.: Development of a model for classification of toxin-induced lesions using ^{1}H NMR spectroscopy of urine combined with pattern recognition. NMR in Biomedicine 11, 235–244 (1998)
5. Fieno, T., Viswanathan, V., Tsoukalas, L.: Neural network methodology for ^{1}H NMR spectroscopy classification. In: Proc. Int. Conf. on Information Intelligence and Systems, pp. 80–85 (1999)
6. Beckonert, O., et al.: NMR-based metabonomic toxicity classification: hierarchical cluster analysis and k-nearest-neighbour approaches. Analytica Chimica Acta 490, 3–15 (2003)
7. Ebbels, T., et al.: Toxicity classification from metabonomic data using a density superposition approach: CLOUDS. Analytica Chimica Acta 490, 109–122 (2003)
8. Lindon, J.C., et al.: Contemporary issues in toxicology the role of metabonomics in toxicology and its evaluation by the COMET project. Toxicology and Applied Pharmacology 187, 137–146 (2003)
9. Specht, D.F.: Probabilistic neural networks. Neural Networks 3, 109–118 (1990)
10. Lienemann, K., Plötz, T., Fink, G.A.: On the application of SVM-Ensembles based on adapted random subspace sampling for automatic classification of NMR data. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 42–51. Springer, Heidelberg (2007)
11. Lienemann, K., Plötz, T., Fink, G.A.: Automatic classification of NMR spectra by ensembles of local experts. In: Structural, Syntactic, and Statistical Pattern Recognition. LNCS, vol. 5342, pp. 790–800. Springer, Heidelberg (2008)
12. Breiman, L.: Stacked regressions. Machine Learning 24, 49–64 (1996)
13. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research 10, 271–289 (1999)
14. dos Santos, E.M., et al.: Overfitting in the selection of classifier ensembles: a comparative study between PSO and GA. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 1423–1424. ACM, New York (2008)

15. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Advances in Large Margin Classifiers, pp. 61–74. MIT Press, Cambridge (1999)
16. Lin, H.T., Lin, C.J., Weng, R.: A note on platt's probabilistic outputs for support vector machines. Machine Learning 68, 267–276 (2007)
17. Matthews, B.W.: Comparison of the predicted and observed secondary structure of the T4 phage lysozyme. Biochimica et Biophysica Acta 405, 442–451 (1975)
18. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
19. Wold, H.: Estimation and Prediciton. In: Estimation of Principal Components and Related Models by Iterative Least Squares, pp. 391–420. Academic Press, London (1966)

Boosting Support Vector Machines Successfully

Kai Ming Ting and Lian Zhu

Gippsland School of Information Technology,
Monash University, Australia
{kaiming.ting,lzhu}@infotech.monash.edu.au

Abstract. Boosting has been shown to improve the predictive performance of unstable learners such as decision trees, but not of stable learners like support vector machines (SVM). In addition to the model stability problem, the high computational cost of SVM prohibits it from generating multiple models to form an ensemble for large data sets. This paper introduces a method that not only enables boosting to improve the predictive performance of SVM, but also reduces the computational cost to make ensembles of SVM feasible for large data sets. The method proposes to build local models, instead of global models; and it is the first method, to the best of our knowledge, to solve the two problems in boosting SVM at the same time. The proposed method to boost SVM also performs better than boosting decision trees in term of predictive accuracy in our experiments.

Keywords: Boosting, support vector machines, local models.

1 Introduction

In recent years, the idea of combining multiple classifiers has attracted a lot of attention. Ensemble approaches like Boosting [1] reduce generalization error of machine learning systems by building and aggregating diverse multiple classifiers.

Although Boosting has been applied to many problems and improved classification accuracy, it suffers from the limitation of its basic theoretically property—it is only suitable to improve predictive performance of unstable learners [2-4]. Boosting is to grow an ensemble of classifiers by successive reweighting of the training set where current weight depends on the previous classifier's performance [5]. It exploits the "instability" of base learners such as decision trees, to build diverse models. Thus Boosting cannot decrease prediction error of stable learners such as Support Vector Machines (SVM). But, stable learners can be used to achieve superior performance for some data sets than unstable learners such as decision trees. For many problems, SVM is preferred for the better match of problems we have on hand.

In addition to the model stability problem, long training time of SVM prohibits it from being applied to large data sets. Although the training time of SVM has been reduced by using Sequential Minimal Optimization (SMO) to empirically between ~m to ~$m^{2.2}$ [6], the high computational cost makes it infeasible to form ensemble of SVM for large data sets.

In this paper we propose a method that not only enables Boosting to improve SVM's prediction accuracy but also reduce the training time of Boosting SVM. The central idea of this approach is to build local models rather than a global model. We called the proposed method ***Local**$_{svm}$*. We show in this paper that this can be achieved by building SVM at every leaf of a decision tree. This increases the instability of the model and at the same time reduces the SVM training time. For example, in the largest data set we used in the experiment, training a ***Local**$_{svm}$* model requires less than one-thousandth of the time used to train one single SVM! And we show that Boosting can be used to successfully improve the predictive accuracy of SVM.

We introduce ***Local**$_{svm}$* in the next section, and describe the evaluation result in Section 3. We discuss the related work in Section 4, and conclude in the last section.

2 Local$_{svm}$

We propose a generic method to learn local models instead of global models such that it converts a stable model into an unstable model and enables existing ensemble methods such as Boosting to be used to improve the stable learner's predictive performance. The key idea is to subdivide the feature space into non-overlapping regions and then use the stable learners to build local models at each of the local regions. As such, the proposed method can be easily implemented using the existing decision tree [8] to construct a non-overlapping subdivision of local regions. By setting an appropriate minimum data size at the leaf for the intended stable learner, it will build a local model most appropriate for each of the local regions. In this paper, we focus on stable learner SVM and use it to build a local model to associate with each leaf in the decision tree; thus we call the proposed method ***Local**$_{svm}$*.

Figure 1 shows an example of ***Local**$_{svm}$*. An internal (non-terminal) node represents a test using one attribute. A leaf (terminal node) represents a local SVM model, trained using a local training set defined by the local region. The learning algorithm is provided in Figure 2.

During classification, a test instance traverses from the root of a tree to a leaf, according to the outcome of the test at each internal node along the path. When it reaches the leaf, classification is made by using the local SVM.

In a nutshell, ***Local**$_{svm}$* is as unstable as a decision tree and yet it classifies instances based on the outcomes of SVM.

The proposed method brings about three key advantages. First, it converts any stable learners into unstable learners so that existing ensemble methods (which apply to unstable learners only) are readily employed to improve the predictive accuracy of the stable learners. We demonstrate the feasibility of the proposed method using the stable learner SVM and an existing ensemble method Boosting in this paper. Second, the model diversity increases significantly through the use of local models as compared a single global model. Third, the use of local training set reduces the training time significantly for high order polynomial time complexity algorithms such as SVM. The analysis of the training time reduction is provided below.

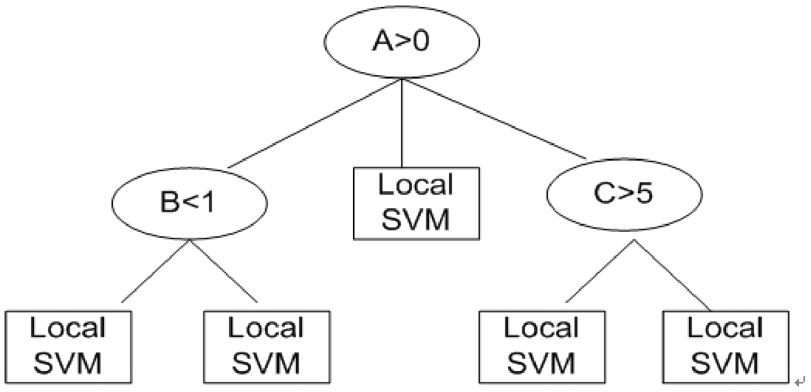


Fig. 1. An example of $Local_{svm}$

```
BuildLocal_svm(S)
Input:   S - Training set
Output:  T - Local_svm

Create a new tree node T
    IF (all instances in the training set S belong to a single class)
    THEN
            make T a leaf with the majority class.
    ELSE IF (number of instances in T is less than 2M)
                                        \* M is the minimum number of instances per leaf. *\
    THEN
          make T a leaf and build a local model associated with this leaf
              T.local_model =  SVM(S)
    ELSE

        /* find the best splitting attribute by using the gain ratio criterion */
              A = selectBestSplitAttribute(S)

              Splitting S into i subsets v_i by using A
              FOR each subset i
                    T.branch(i) = BuildLocal_svm (v_i).
              END FOR

    END IF

RETURN T
```

Fig. 2. The $\boldsymbol{Local_{svm}}$ agorithm

Time complexity analysis. Assume SVM used in ***Local**$_{svm}$* has time complexity $O(m^p)$, where m is the number of training instances and p is a constant. Let f be the number of leaf nodes in a decision tree in which f local models of SVM are trained, and further assume that the training data is distributed uniformly over f. Thus, the ratio of reduction in training time (T_R) due to ***Local**$_{svm}$* is given as follows:

$$T_R = \frac{m^p}{f\left(m/f\right)^p} = f^{\,p-1}$$

As a result, the time reduction due to ***Local**$_{svm}$* as compared to SVM can be significantly when both f and p are large. As an example, in two of the most time consuming data sets (connect and coding), training a single SVM model took 16000 seconds and 1100 seconds, respectively; whereas training a single ***Local**$_{svm}$* (having f = 403 and 91, respectively) took only 14 seconds and 4 seconds, respectively using the same machine.

Knowing f and the training time reduction ratio in real experiments, we can compute p based on the above equation: p is calculated to be 2.2 for both data sets—the upper end of the time complexity estimated by Platt using the SMO algorithm [6].

3 Empirical Study

The empirical study has two aims. First, to examine whether SVM can be successfully boosted using ***Local**$_{svm}$*. Here we investigate Boost ***Local**$_{svm}$* in terms of its predictive accuracy and its capacity to handle large data sets in terms of training time. Second, we compare Boost ***Local**$_{svm}$* to the commonly used combination of boosting and decision tree, an unstable learner. The experiments are carried out in the Weka platform [7]. We implement ***Local**$_{svm}$* in Weka by using J48 (Weka's implementation of C4.5 decision tree [8]). The minimum number of instances at each leaf is set to 100 for ***Local**$_{svm}$*. We use the Weka implementations of AdaBoost.M1 and SMO for boosting [1] and SVM [6] with linear kernels in the experiments. We use seven large data sets from the UCI Repository [9] in the experiments. Because of the large data sizes, only a single run is carried out in each data set using the given training and test sets. The characteristics of those data sets are shown in Table 1.

Table 1. Description of data sets used in the experiments

	#Numeric attribute	#Normal attribute	#Classes	Train Size	Testing Size
connect	0	42	3	50000	17557
poker-hand	0	10	9	25010	1000000
coding	0	15	2	15000	5000
nursery	0	8	5	8606	4354
letter	16	0	26	16000	4000
shuttle	9	0	7	43500	14500
pendigits	16	0	10	7494	3489

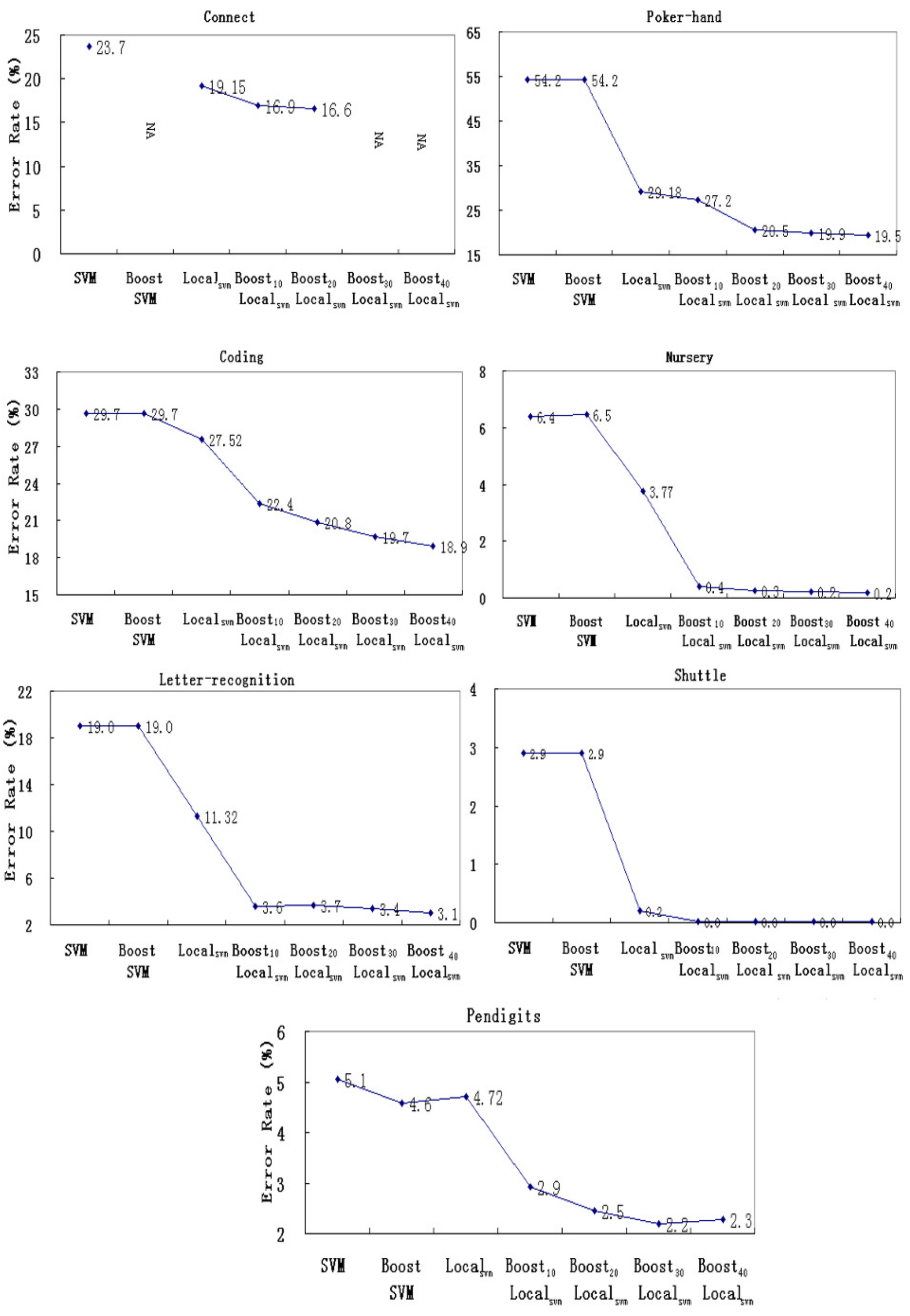


Fig. 3. Error rate results for SVM, Boost SVM, ***Local$_{svm}$*** and Boost$_i$ ***Local$_{svm}$***, where i is the number of boosting iterations

The following description is divided into two subsections. The first subsection describes the result of the investigation comparing SVM, Boost SVM, ***Local**$_{svm}$* and Boost ***Local**$_{svm}$*. The second subsection shows the results comparing Boost ***Local**$_{svm}$* and Boost J48. All the experiments are run in AMD 2356 Quad Core Opterons™, 32GB RAM and 4 x 320GB disks with CentOS 5 Linux operating system.

3.1 Can SVM Be Successfully Boosted Using *Local*$_{\mathrm{svm}}$?

Fig. 3 shows the error rate result for SVM, Boost SVM, ***Local**$_{svm}$*, Boost ***Local**$_{svm}$* using 10, 20, 30 and 40 iterations in the seven data sets. As expected, the result shows that boosting cannot improve the predictive performance of SVM because of its model stability. Only in the pendigits data set that Boost SVM shows some marginal improvement. On the other hand, ***Local**$_{svm}$* alone improves the predictive accuracy of SVM significantly; and Boost ***Local**$_{svm}$* further improves its predictive accuracy and the improvement continues with the increased number of iterations. The only exception is in the pendigits data set in which the error rate continues to decrease up to 30 boosting iterations but it increases slightly when the boosting iterations reach 40.

Fig.4 shows the training time result for SVM, Boost SVM, ***Local**$_{svm}$* and Boost ***Local**$_{svm}$*. For data sets in which SVM requires long training time, ***Local**$_{svm}$* significantly reduces the training time. For example, in the connect, poker-hand and coding data sets, training one ***Local**$_{svm}$* model only requires one-thousandth, one-eightieth and one-three-hundredth of the time required to train one SVM model! As a result, training 40 ***Local**$_{svm}$* models (in Boost ***Local**$_{svm}$* with 40 iterations) requires less time than that to train a single model SVM model in these data sets!

In data sets in which SVM trains fast, ***Local**$_{svm}$* will need training time in the same order. The examples can be found in the letter-recognition, shuttle and pendigits data sets.

In all data sets, Boost ***Local**$_{svm}$* demonstrates that the training time increases linearly with the number of iterations.

It is important to note that it will takes too long to run Boost SVM in the connect data set when training a single SVM model takes more than 4.5 hours; boosting 40 SVM models will take more than 7 days! Also note that Boost SVM usually takes significantly longer to run than Boost ***Local**$_{svm}$* when the same number of iterations is used. For example, Boost SVM with 49 iterations takes more than 900 seconds in the nursery data set; but Boost ***Local**$_{svm}$* with 40 iterations takes less than 40 seconds only. In the coding data set, boosting 2 SVM models takes more than 2800 seconds; whereas boosting 40 ***Local**$_{svm}$* models takes less than 130 seconds only.

3.2 Comparison with J48 and Boost J48

Table 2 shows the comparison with J48 and Boost J48. First, ***Local**$_{svm}$* performs better than J48 in five out of the seven data sets in terms of predictive accuracy; the two exceptions are marginally worse. Second, Boost ***Local**$_{svm}$* performs better than Boost J48 in majority of the data sets and performs equally well in one to two data sets (depending on the number of iterations used.) The only exception is in the coding data set in which Boost ***Local**$_{svm}$* performs marginally worse when boosting 10 iterations and becomes marginally better when boosting 40 iterations.

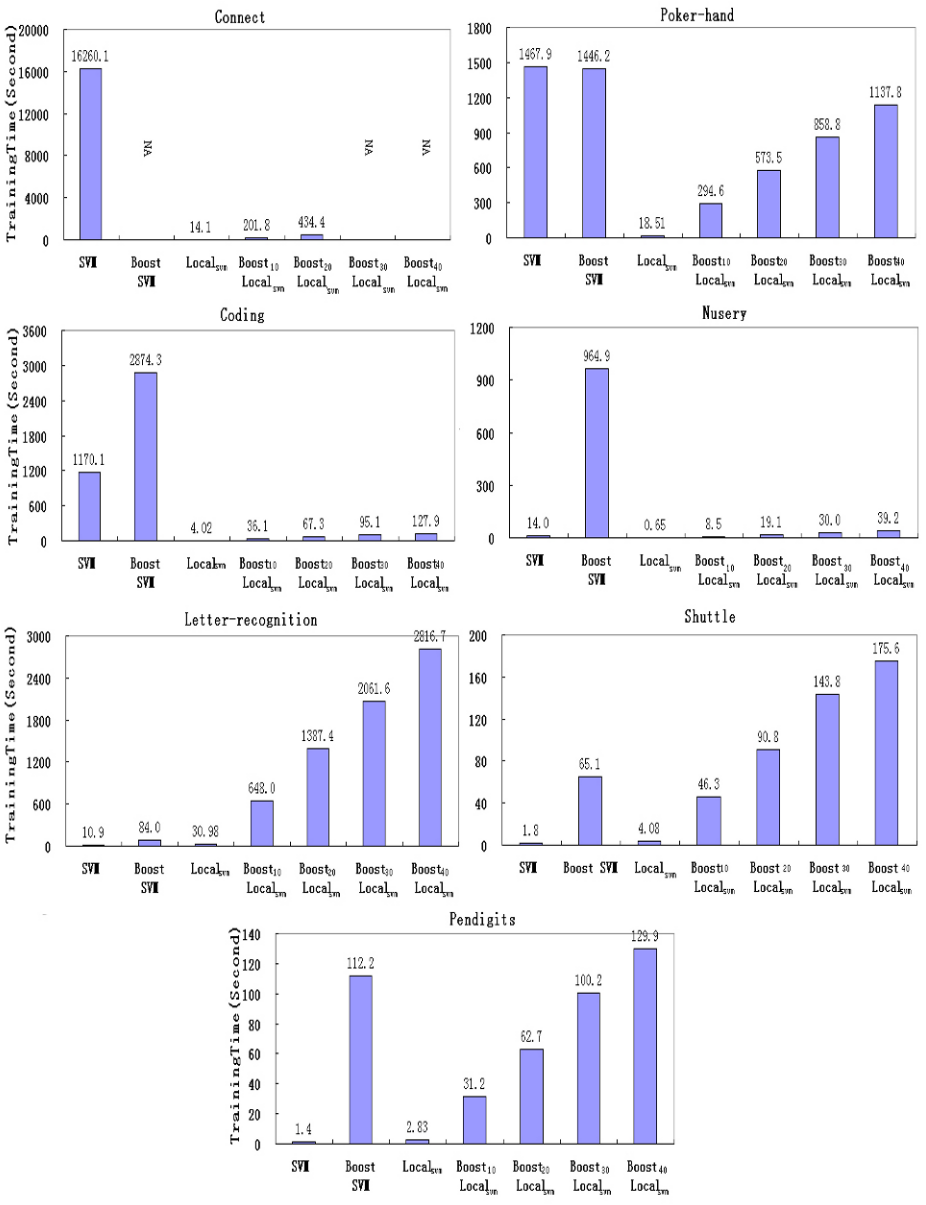


Fig. 4. Training time result for SVM, Boost SVM, ***Local***$_{svm}$ and Boost ***Local***$_{svm}$. Boost SVM stops at 1, 2, 49, 4, 4, and 21 iterations respectively in the poker-hand, coding, nursery, letter, shuttle, and pendigits data sets.

In terms of training time, it is expected that ***Local***$_{svm}$ will take longer because of the use of SVM at every leaf of the tree. Indeed, Boost ***Local***$_{svm}$ is either in the same order or requires 1-2 order more training time than Boost J48.

Table 2. Result of error rate and training time for J48, ***Local**$_{svm}$* and their boosting counterparts

Error Rate (%)				**Boost 10**		**Boost 40**	
	J48	**SVM**	***Local**$_{svm}$*	**Boost J48**	**Boost *Local**$_{svm}$*	**Boost J48**	**Boost *Local**$_{svm}$*
connect	19.66	23.7	19.15	17.4	16.9	NA	NA
poker-hand	42.3	54.2	29.18	36.9	27.2	30.7	19.5
coding	29.2	29.7	27.52	22.2	22.4	19.1	18.9
nursery	3.7	6.4	3.77	1.3	0.4	0.7	0.2
letter-recog	12.47	19.0	11.32	4.8	3.6	3.1	3.1
shuttle	0.05	2.9	0.2	0.0	0.0	0.0	0.0
pendigits	7.95	5.1	4.72	3.7	2.9	3.0	2.3
Training Time (sec)							
connect	3.74	16260.1	14.1	46.2	201.8	NA	NA
poker-hand	0.45	1467.9	18.51	4.0	294.6	15.9	1137.8
Coding	0.43	1170.1	4.02	2.6	36.1	10.6	127.9
Nursery	0.15	14.0	0.65	0.8	8.5	3.1	39.2
letter-recog	4.6	10.9	30.98	44.3	648.0	185.7	2816.7
Shuttle	4.25	1.8	4.08	33.9	46.3	141.6	175.6
Pendigits	0.93	1.4	2.83	8.2	31.2	32.3	129.9

4 Related Work

Boosting has previously been applied to SVM with the sole purpose of either scaling up the training time [10] or increasing predictive accuracy [18]. Boosting is used in [10] to select a subsample so that a SVM can be trained using a smaller subset. In [18], the boosting procedure is restarted, when the termination condition is met, for different values of the kernel parameter in order to increase model diversity—this increases the training time of a normal boosting process without reducing the training time for individual SVM—this is infeasible for large data sets. In another work, bagging is used in conjunction with random subspacing to increase SVM's instability with the aim to improve predictive accuracy without significantly reduction in the training time [11]. In contrast, ***Local**$_{svm}$* enables SVM to be successfully boosted and its training time to be significantly reduced, especially in large data sets.

An ensemble built using ***Local**$_{svm}$* is different from that built using methods such as Random Subspace [17] in three key aspects. First, Random Subspace builds each global model using all data points, albeit with one reduced feature set; whereas ***Local**$_{svm}$* builds local models using significantly small local training sets, each with a different reduced feature set. Second, the diversity of the models generated by Random Subspace is limited to the set of C^n_h feature subsets, where C^n_h is the binomial coefficient and h is the number of selected features out of the total n features. Each ***Local**$_{svm}$* model has f different local models trained using different feature subsets as opposed to only one feature subset used in one Random Subspace model. Diversity of ***Local**$_{svm}$* models is further enhanced by the ensemble method used. Third, for an algorithm with $O(nm^p)$, the training time for Random Subspace is expected to be reduced by half only, when $h = n/2$

is used in order to have the maximum diversity; whereas the training time for ***Local**$_{svm}$* is expected to be reduced less than $nm^p/f^{\,(p-1)}$, since each local model is trained using a reduced feature set.

To the best of our knowledge, the only other ensemble approach to improve the predictive performance of stable learners is Davidson's bootstrap model averaging [12] which creates global models from multiple bootstrapping samples and sums the joint probabilities of individual ensemble members to produce the final prediction (as opposed to a simple majority vote in bagging.) This method has two key weaknesses. First, the diversity of the models depends on bootstrap samples only; as a result, the improvement over, bagging due solely to model averaging, is small. Second, because of the use of global models, no significant training time reduction can be expected; thus, it cannot be applied to high order polynomial time complexity algorithms, especially in large data sets.

Recent progress has significantly reduced the time complexity of SVM to $O(m)$ by using Core Vector Machines [13] and Ball Vector Machines [14]. Despite these advances, they are still the stable learners which prohibit SVM from being successfully boosted. These methods can be used in conjunction with ***Local**$_{svm}$* to improve the training time in ensembles.

There are previous works which build hybrid models using decision trees in the same style as suggested in this paper. For example, Ting and Zhang [15] study the weaknesses of boosting Naïve Bayes by building Naïve Bayes at each leaf of a decision tree; Kohavi [16] scales up the predictive accuracy of a single Naïve Bayes using such a hybrid. But none has used SVM in a decision tree hybrid we have proposed in this paper and showed that it can be successfully boosted.

5 Conclusions and Future Work

We show in this paper that one can successfully boost SVM by building local SVMs at the leaves of a decision tree. This resolves the model stability problem and the high computational cost problem at the same time, and makes boosting SVM a reality that would otherwise infeasible for large data sets. To the best of our knowledge, this is the first method to successfully boost SVM which improves the predictive accuracy of the base model as well as significantly reduce the training time for SVM. We also show in our experiments that the proposed method to boost SVM perform better than boosting decision trees in terms of predictive accuracy.

Although we have focused in this paper on boosting SVM, the proposed method which advocates the use of local models (as opposed to global models), we believe, can be applied to other stable learners, and the base model created is readily applied to existing ensemble methods such as Bagging, Random Subspace which rely on unstable models, not just for Boosting only. We will verify this in the near future.

References

1. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148–156 (1996)
2. Zenobi, G., Cunningham, P.: Using Diversity in Preparing Ensembles of Classifiers Based on Different Feature Subsets to Minimize Generalization Error. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS, vol. 2167, pp. 576–587. Springer, Heidelberg (2001)

3. Kuncheva, L.I., Whitaker, C.J.: Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. Machine Learning 51, 181–207 (2003)
4. Lam, L.: Classifier Combinations. Implementations and Theoretical Issues. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 77–86. Springer, Heidelberg (2000)
5. Freund, Y.: An adaptive version of the boost by majority algorithm. In: Proceedings of the 12th Conference on Computational Learning Theory, pp. 102–113. ACM Press, New York (1999)
6. Platt, J.C.: Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical Report MST-TR-98-14, Microsoft Research (1998)
7. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
8. Quinlan, J.R.: C4. 5: programs for machine learning. Morgan Kaufmann, San Francisco (1993)
9. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository, University of California, Irvine, CA (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
10. Pavlov, D., Mao, J., Dom, B.: Scaling-up support vector machines using the boosting algorithm. In: International Conference on Pattern Recognition, pp. 219–222 (2000)
11. Tao, D., Tang, X., Li, X., Wu, X.: Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. IEEE Transactions in Pattern Analysis and Machine Intelligence 28(7), 1088–1099 (2006)
12. Davidson, I.: An Ensemble Technique for Stable Learners with Performance Bounds. In: Proceedings of the 19th National Conference on Artificial Intelligence, pp. 330–335 (2004)
13. Tsang, I., Kwok, J.T., Cheung, P.-M.: Core Vector Machines: Fast SVM Training on Very Large Data Sets. Journal of Machine Learning Research 6, 363–392 (2005)
14. Tsang, I.W., Kocsor, A., Kwok, J.T.: Simpler core vector machines with enclosing balls. In: Proceedings of the 24th International Conference on Machine learning, pp. 911–918 (2007)
15. Ting, K.M., Zheng, Z.: A Study of AdaBoost with Naïve Bayesian Classifiers: Weakness and Improvement. Computational Intelligence 19(2), 186–200 (2003)
16. Kohavi, R.: Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. In: Proceedings of the 2nd KDD, pp. 202–207 (1996)
17. Ho, T.K.: The Random Subspace Method for Constructing Decision Forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
18. Li, X., Wang, L., Sung, E.: A Study of AdaBoost with SVM based Weak Learners. In: Proceedings of the International Joint Conference on Neural Networks, pp. 196–201 (2005)

Manifold Learning for Multi-classifier Systems via Ensembles

Melba Crawford and Wonkook Kim

Laboratory for Applications of Remote Sensing
Purdue University
mcrawford@purdue.edu, wkkim@purdue.edu

Abstract. Statistical classification of hyperspectral data is challenging because the inputs are high in dimension, while the quantity of labeled data is typically limited. The resulting classifiers are often unstable and have poor generalization. Nonlinear manifold learning algorithms assume that the original high dimensional data actually lie on a low dimensional manifold defined by local geometric differences between samples. Recent research has demonstrated the potential of these approaches for nonlinear dimension reduction and representation of high dimensional observations. Nonlinear scattering phenomena associated with processes observed in remote sensing data suggest that these may be useful for analysis of hyperspectral data. However, computational requirements limit their applicability for classification of remotely sensed data. Multi-classifier systems potentially provide a means to exploit the advantages of manifold learning through decomposition frameworks, while providing improved generalization. This paper reports preliminary results obtained from an ensemble implementation of Landmark Isomap in conjunction with a kNN classifier. The goal is to achieve improved generalization of the classifier in analysis of hyperspectral data in a dynamic environment with limited training data. The new method is implemented and applied to Hyperion hyperspectral data collected over the Okavango Delta of Botswana.

1 Introduction

Increased availability of data from airborne and space-based hyperspectral sensors has generated tremendous interest in the remote sensing community. These instruments characterize spectral signatures with much greater detail than traditional multispectral sensors, and thereby can potentially provide improved discrimination of targets [1]. However, hyperspectral data also present difficult challenges for supervised statistical classification, where labeled training data are used to estimate the parameters of the label-conditional probability density functions [2]. Often, the dimensionality of the data is high (>200), there are tens of classes C, and the quantity of training data is small. Sample statistics of training data may also not be representative of the true probability distributions of the individual class signatures, particularly for remote, inaccessible areas where training data are logistically difficult and expensive to acquire. Generalization of the resulting classifiers is often poor, thereby resulting in poor quality mapping over extended areas.

Various approaches have been investigated to mitigate the impact of small sample sizes and high dimensionality, which are inherently coupled issues since the adequacy of a data sample depends on the data dimensionality, among other factors [3]. Regularization methods attempt to stabilize the covariance matrix by weighting the sample covariance matrix and a pooled covariance matrix or by shrinking the sample covariance matrix toward the identity matrix [4]. While this may reduce the variance of the parameter estimates, the bias of the estimates can increase dramatically. Alternatively, the input space can be transformed into a reduced feature space via feature selection [5,6] or feature extraction[7,8,9]. Although feature selection methods reduce the effect of the high dimensionality problem and are more interpretable, they are often trapped in a local optimal feature subset. Linear feature extraction methods, including principal component analysis (PCA), the maximum noise fraction (MNF), decision boundary feature extraction (DBFE) [7], segmented principal component analysis (SPCA) [8], and best basis band combining methods [9,10] are all used in analysis of hyperspectral data. While these methods have been successful in many classification problems, they ignore the nonlinear scattering phenomena represented in the bidirectional reflectance distribution fraction (BDRF). Multiple scattering within pixels, mixed pixels, atmospheric variability, scene geometry, and canopy characteristics all contribute to nonlinear responses inherent in hyperspectral data [11], motivating investigation of nonlinear methods.

2 Methodology

Recently, the machine learning community has made significant progress on modeling nonlinear structure by determining coordinate systems that lie on the nonlinear manifold represented by the data [12,13,14,15,16]. The connection of two of the most popular methods, Isomap [12] and Local Linear Embedding [13] to kernel PCA [17] imply that new data can also be readily projected onto the manifold. This is an important advance for classification problems as the manifold is often constructed from training data, and the novel observations must be projected onto the manifold.

Isomap nonlinear manifold learning assumes that the local feature space formed by the nearest neighbors is linear, and the global nonlinear pattern can be found by connecting these piecewise linear spaces. Isomap uses a user-defined neighborhood and the shortest path algorithm to discover the manifold. It first defines K_i, the set of neighborhood nodes of node i, to create a distance matrix $\mathbf{D}'$. If $j \in K_i$, $d'_{ij} = d_{ij}$. If $j \notin K_i$, $d'_{ij} = \infty$. Isomap then accumulates the distance beyond the set K_i along the shortest path to obtain $\mathbf{D}_{stp}$.

The shortest path network is constructed from a directed graph $G = (N, E)$, where N represents the nodes, and E represents the edges of the graph. The value of d_{ij} represents the length (cost) of E_{ij}, while x_{ij} is the amount of flow from N_i to N_j. The shortest path algorithm, typically implemented in Isomap via the Dijkstra [18] method, finds the paths from a root node N_1 to all other nodes to minimize the sum of the individual path lengths. The process is repeated for each sample, which in turn becomes the root node, to create a shortest path network $\mathbf{D}_{stp}$.

Dimension reduction is then accomplished through multidimensional scaling (MDS), a linear dimension reduction technique that places a set of samples in a meaningful dimensional space that explains the similarity between samples. Given a distance matrix $\mathbf{D}$, and assuming that a $\mathbf{Y} \in \Re^{l \times n}, l \ll d$ exists such that $\delta_{ij}^2 = ||\mathbf{y}_i - \mathbf{y}_j||^2 \approx d_{ij}^2$ and $\mathbf{Y}_i$ are orthogonal, it can be shown that $\mathbf{Y}$, calculated by classical MDS, is equivalent to a vector of the first l principal components of $\mathbf{X}$ if the Euclidean pairwise distance matrix is used [19]. Here, MDS is used to evaluate the instrinsic dimension of $\mathbf{D}_{stp}$.

Experiments in [12] have demonstrated that $\mathbf{D}_{stp}$ is able to define the nonlinear manifold, and that it can be represented globally by MDS in a lower dimensional space. The method has been applied to hyperspectral data by researchers at the Naval Research Laboratory and by our group [11,20,21]. Both research groups found that Isomap can characterize hyperspectral data in low dimensional spaces while improving separation for many classes, so the embedded features are potentially useful for classification. However, although Dijkstra's algorithm is efficient for finding the shortest path from a root node to the rest of the nodes, building the shortest path network is $O(N^2 \log N)$, which is problematic when the total number of samples, N, is large.

Several approaches have been proposed to mitigate the computational demands of Isomap. Bachmann [11] approached the memory and computational overhead problem by dividing the scene into arbitrary subsets, each with a manageable number of samples, and realigning them after learning the individual manifolds. In another approach, Landmark Isomap (L-Isomap) randomly selects n "landmark" points from the original data to construct its manifold [16]. Instead of building an $N \times N$ shortest path network, L-Isomap uses a much smaller $n \times N$ network, which requires fewer iterations. The MDS operations are also reduced on this smaller network. Samples that are not selected for landmarks are placed on the manifold via the derived embedding vectors and their updated distances to the n landmark points. The complexity of computing the geodesic distance matrix is then $O(nN \log n)$. Although it mitigates the computational burden, L-Isomap assumes that the manifold is smooth, which is not always the case for data in remotely sensed images. Chen et al. [20] investigated classification with manifold learning by using the kNN method. Later, they obtained improved classification results using a new form of landmark selection which focuses on the boundaries of the clusters, rather than randomly selected points as in L-Isomap [22]. In essence, the goal was to identify the facets of the manifold that are often associated with boundaries of the classes.

Extending these ideas, Bachmann et al. [21] investigated use of a representative backbone manifold composed of a subset of samples, and subsequent embedding of other samples in the manifold via a local linear embedding algorithm.

Using a different approach, Kim et al. [23] sought to incorporate local spatial information and unlabeled samples while reducing computation associated with developing the Isomap manifold by performing spatial-spectral clustering prior to developing multiresolution manifold in a two-stage approach. Later, they also investigated Laplacian regularization as a semi-supervised approach to mitigate the impact of small training sets [24].

While these approaches have all improved computation in manifold learning, new improvements may also be achieved through multi-classifier systems. Ensembles of manifolds and decomposition methods such as binary and hierarchical classifiers, which also provide capability to exploit local manifold structure, all provide potentially advantageous frameworks for reducing computational overhead, while increasing generalization of classifiers. In this preliminary study, Isomap is investigated for manifold learning in an ensemble-based approach, whereby labeled data are severely decimated by random sampling to create multiple sparse approximations to the full manifold.

The goal of the research is to investigate the capability of a multi-classifier system based on an "extreme" implementation of L-Isomap, whereby a dramatically reduced number of points is used to construct each component of an ensemble of manifolds. Novel pixels are classified relative to the resulting manifolds. While individual manifolds obtained using the new Ensemble L-Isomap (EL-Isomap) are not likely to be representative approximations of all classes, the ensemble of manifolds may be adequate to achieve good classification results. The computational complexity of EL-Isomap increases linearly with the number of components in the ensemble, so the overall computational requirements are greatly reduced.

Classification in this study is performed using the kNN method, which is easy to implement with Isomap since both require the distance matrix of the data. Because Isomap obtains geodesic distances from the distance matrix by calculating the shortest path distances, kNN classification can be applied once the matrix is obtained. Thus, the manifold coordinates are not actually obtained, mitigating the impact of selection of the intrinsic dimension, to which manifold based classification is quite sensitive. Finally, EL-Isomap determines the overall label of a pixel via voting.

3 Results

The ensemble approach for classification was applied to hyperspectral data and evaluated relative to classification accuracies obtained by the traditional Isomap and the L-Isomap methods.

3.1 Remotely Sensed Data

The NASA EO-1 satellite acquired a sequence of data over the Okavango Delta, Botswana in 2001-2003. The Hyperion sensor on EO-1 acquires data at 30 m^2 pixel resolution over a 7.7 km strip in 242 bands covering the 400-2500 nm

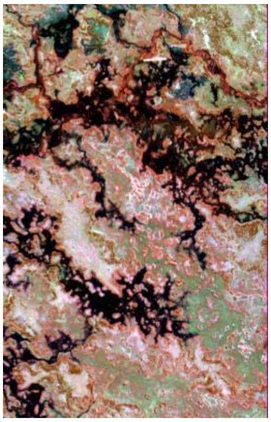


Fig. 1. Subset of Hyperion data over the Okavango Delta (RGB: Bands 51, 149, 31)

Table 1. Land cover classes for Okavango Delta region and associated numbers of labeled data

Land Cover Classes		# Labeled	Disjoint Labeled Data Sets	
		Data Points	Data 1	Data 2
1	Water	297	158	139
2	Primary Floodplain	437	228	209
3	Riparian	448	237	211
4	Firescar	354	178	176
5	Island Interior	337	183	154
6	Woodlands	357	199	158
7	Savanna	330	162	168
8	Short Mopane	239	124	115
9	Exposed Soils	215	111	104
	TOTAL	3014	1580	1434

portion of the spectrum in 10 nm windows. Preprocessing of the data was performed to mitigate the effects of bad detectors, inter-detector miscalibration, and intermittent anomalies. Uncalibrated and noisy bands associated with water absorption features were removed, and the remaining 145 bands [10-55, 82-97, 102-119, 134-164, 187-220] were included as candidate features. Figure 1 illustrates the complexity of the spatial distribution of the data. A data set comprised of 1476 $\times$ 256 pixels with nine land cover types (Table 1) was analyzed in this study. Labeled data were obtained from a combination of field studies and interpretation of high resolution imagery by experts who are knowledgeable of the area. Because land cover types are collocated in geographic regions, labeled data occur in contiguous patches. Training and test data were obtained by randomly sampling data from these patches. A set of spatially disjoint labeled data were also collected throughout the scene to better evaluate generalization of the method.

3.2 Experimental Design

Experiments were conducted to evaluate the impact of sampling rate, number of elements in the ensemble, and the effect of incorporating unlabeled data in construction of the manifold when the quantity of labeled data is extremely small. Results were obtained for manifolds constructed using I) randomly selected subsets of all the labeled data, II) randomly sampled labeled data from disjoint areas, III) labeled and unlabeled data from disjoint areas. Hereafter, these experiments are referred to as I, II and III, respectively.

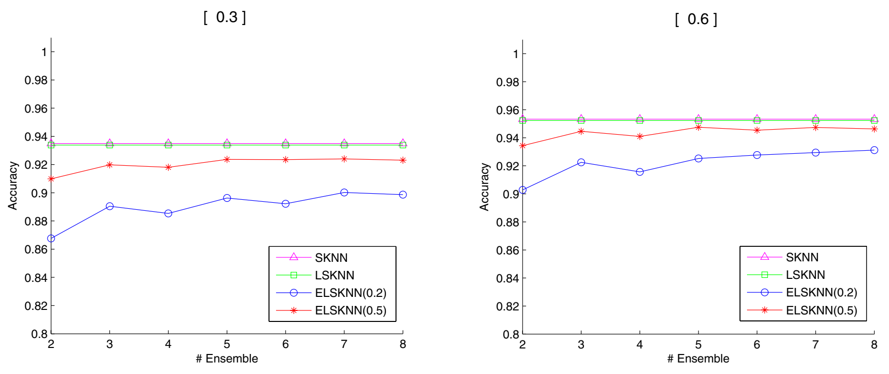
Data were randomly sampled such that 25% were reserved for testing, and the remaining points were sampled to obtain training sets comprised of 15%, 30%, 45%, 60%, and 75% of the original data. These training sets were used to construct the manifolds for Isomap and L-Isomap experiments. For EL-Isomap, the manifolds were constructed by further decimating the L-Isomap training sets to 20%, 30%, 40%, and 50% of the labeled samples utilized for L-Isomap (e.g. 20×15 samples, etc). Ten replications of each split of the training and test data and 5 replications of each EL-Isomap experiment were performed, respectively. The optimal value of k for the kNN classifier was obtained by 3-fold cross validation. The number of neighbors used in construction the shortest path network was selected experimentally to be 5.

3.3 Experimental Results

Complete results obtained for all sampling rates are too extensive to be presented herein. Representative sample results and a summary of trends exhibited in the full set of experimental results are included. Plots of the classification accuracies obtained using the kNN classifier in conjunction with Isomap (SKNN), traditional L-Isomap (LKNN), and Ensemble Landmark Isomap (ELSKNN) are shown in Fig. 2- 4.

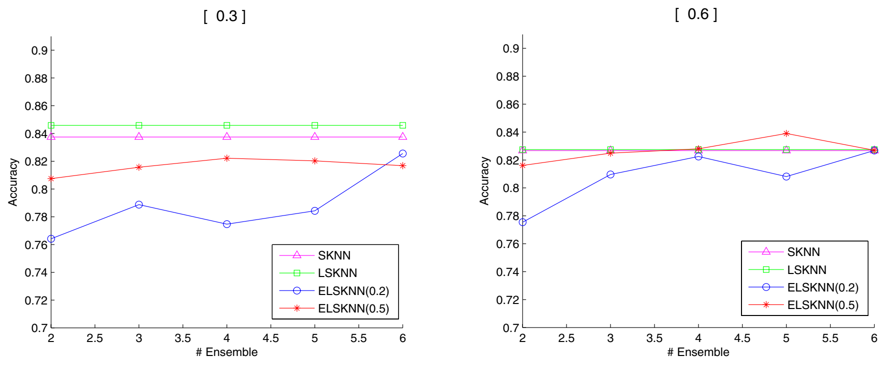
Classification results obtained using SKNN reflect the manifold constructed from the combined set of sampled training and the full test data, while results for LSKNN were obtained from a manifold constructed using only the randomly sampled training data. Thus, for low sampling rates, the accuracies obtained from SKNN and LSKNN are almost identical. As the sampling rate increases, SKNN yields higher accuracies resulting from more points being used to construct the manifold than for LSKNN. The primary focus of the study is the impact of severe decimation of the data used to construct the manifold for the ensemble, so discussion focuses on results from LSKNN and ELSKNN.

The test set is 25% of the original labeled data for Experiment I, approximately 90 points for most classes. At a sampling rate of 30% ($\approx$ 100 points), the manifold for SKNN is constructed from $\approx$ 190 points, while the manifold for LSKNN is constructed from $\approx$100 points. The overall classification accuracies for SKNN and LSKNN are both $\approx$.93, as shown in Fig. 2a. When the landmark sample is further decimated to 20% of the landmark sample ($N \approx 100 \times .2$) for ELSKNN, the average accuracy for an ensemble manifold with $M = 2$ elements is .87, then improves to .89 when $M = 3$ and to .90 when $M = 5$, with little additional improvement for larger M. For a decimation rate of 50% ($N \approx 100 \times .5$)



(a) (b)

Fig. 2. Accuracy plot for Experiment I. SKNN, LSKNN and ELSKNN for two different training rates, (a) 30% and (b) 60%. For ELSKNN, two different decimation rates, 20% and 50%, are shown.



(a) (b)

Fig. 3. Accuracy plot for Experiment II. SKNN, LSKNN and ELSKNN for two different training rates, (a) 30% and (b) 60%. For ELSKNN, two different decimation rates, 20% and 50%, are shown.

for ESKNN, the accuracy is .91 with a 2 element ensemble and .92 for a 3 element ensemble, with little improvement for larger ensembles. Fig. 2b shows the same trends for sampling rates of 60% for the training data. Reduction in computation is significant, with little degradation in classification accuracy relative to LSKNN.

Results from Experiment II obtained using the disjoint labeled data sets have lower accuracies, reflecting the high correlation between training and test data obtained by sampling data that are collocated within a geographic region. Approximately half the data lie in each of the disjoint sets of labeled data, so only about half the number of points ($\approx$ 50 points for a sampling rate 30%, $\approx$ 95

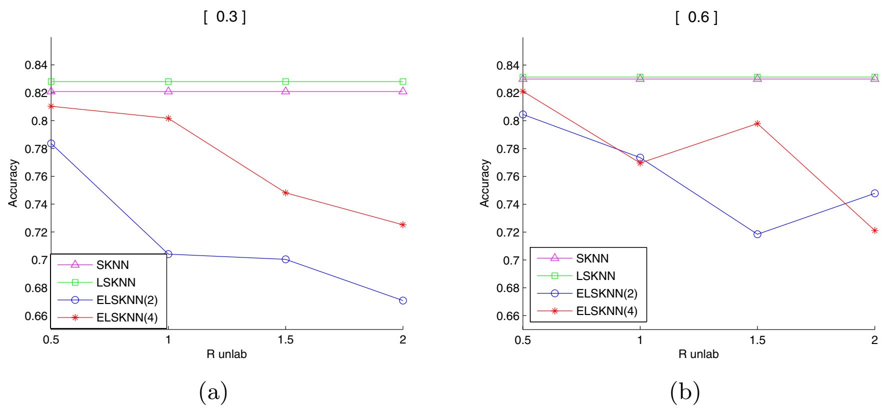


Fig. 4. Accuracy plot for Experiment III. SKNN, LSKNN and ELSKNN for two different training rates, (a) 30% and (b) 60%. For ELSKNN, two different ensemble sizes of 2 and 4 are shown.

points for 60%) is used to construct the manifolds in Fig. 3. The accuracies for a sampling rate of 30% are shown in Fig. 3a, with decimation rates of 20% and 50% for ELSKNN, and Fig. 3b contains accuracies for a sampling rate of 60% and decimation rates of 20% and 50% of the training data. For sampling rates of 30%, the average classification accuracy for LSKNN is .84, while ELSKNN yields an accuracy of .76 for a 2 member ensemble, with only slight improvement achieved with larger ensembles with decimation of 20% ($N \approx 50 \times .2$). With decimation of 50% ($N \approx 50 \times .5$), ELSKNN has nearly the same average accuracy as LSKNN for a 2 member ensemble, but with half the data. When the sampling rate increases to 60% in Fig. 3b ELSKNN has nearly the same accuracy as LSKNN for a decimation rate of 20% ($N \approx 95 \times .2$) with a 2 element ensemble, and actually has higher accuracies than LSKNN for a 5 member ensemble with a decimation rate of 50%.

When the quantity of labeled data are extremely limited, the manifold may not be characterized adequately for good classification. Preliminary experiments conducted to investigate the impact of inclusion of unlabeled data are shown in Fig. 4. Here, the training data were sampled, then augmented by unlabeled data to evaluate the impact of ensemble manifolds. Fig. 4a demonstrates the impact of augmenting the unlabeled data (30% sampling rate, 30% decimation rate) such that the ratio of the number of unlabeled to labeled data ranges from .5 to 2.0 for a 2, 3, and 5 member ensembles. Fig. 4b contains analogous results where the sampling rate is 60%. Classification accuracies degrade as the quantity of unlabeled data introduced to the system increases, but the impact is mitigated by large ensembles. Although the overall accuracies are lower, the accuracies for individual classes may actually improve due to the manifold having been developed from a larger, more representative sample.

4 Conclusions and Ongoing Work

The primary goal of this research is to investigate multiple classifier system frameworks in conjunction with manifold learning for analysis of high dimensional remote sensing image data. Because of the computational complexity of algorithms such as Isomap, application of these approaches to large data sets commonly encountered in hyperspectral sensing is impossible. For classification purposes, manifolds are typically constructed from labeled data, then unlabeled data are inserted for classification. In this paper, we explored the concept of severe decimation of the training data used to build the manifold, referring to the process as "extreme landmark" selection, in conjunction with ensembles. The results indicate that for this data set, lower accuracies associated with significant reduction in the amount of data used to construct the manifold are largely compensated for by the use of ensembles. In practice, the rate of decimation which could be tolerated would need to be ascertained.

Use of multi-classifier systems in conjunction with even simplistic sampling of the data to create data manifolds appears to be promising. Ongoing work involves investigation of other schemes for landmark selection, alternative classifiers, and hierarchical schemes for developing sub-manifolds with class-dependent characteristics.

References

1. Pearlman, J.S., Berry, P.S., Segal, C.C., Shapanski, J., Beiso, D., Carman, S.L.: Hyperion: a space-based imaging spectrometer. IEEE Transactions on Geoscience and Remote Sensing 41(6), 1160–1173 (2003)
2. Landgrebe, D.: Hyperspectral image data analysis. IEEE Signal Processing Magazine 19(1), 17–28 (2002)
3. Raudys, S.J., Jain, A.K.: Small sample size effects in statistical pattern recognition: recommendations for practitioners. IEEE Transactions on Pattern Analysis and Machine Intelligence 13(3), 252–264 (1991)
4. Tadjudin, S., Landgrebe, D.A.: Covariance estimation with limited training samples. IEEE Transactions on Geoscience and Remote Sensing 37(4), 2113–2118 (1999)
5. Serpico, S.B., Bruzzone, L.: A new search algorithm for feature selection in hyperspectral remote sensing images. IEEE Transactions on Geoscience and Remote Sensing 39(7), 1360–1367 (2001)
6. Korycinski, D.: Investigating the use of Tabu Search to find near-optimal solutions in multiclassifier systems. Ph.D thesis, University of Texas at Austin (2003)
7. Lee, C., Landgrebe, D.A.: Decision boundary feature extraction for neural networks. IEEE Transactions on Neural Networks 8(1), 75–83 (1997)
8. Jia, X., Richards, J.A.: Segmented principal components transformation for efficient hyperspectral remote-sensing image display and classification. IEEE Transactions on Geoscience and Remote Sensing 37(1), 538–542 (1999)
9. Kumar, S., Ghosh, J., Crawford, M.M.: Best-bases feature extraction algorithms for classification of hyperspectral data. IEEE Transactions on Geoscience and Remote Sensing 39, 1368–1379 (2001)

10. Morgan, J.T., Henneguelle, A., Crawford, M.M., Ghosh, J.: Adaptive feature spaces for land cover classification with limited ground truth. International Journal of Pattern Recognition and Artificial Intelligence 18(5), 777–800 (2004)
11. Bachmann, C.M., Ainsworth, T.L., Fusina, R.A.: Exploiting manifold geometry in hyperspectral imagery. IEEE Transactions on Geoscience and Remote Sensing 43(3), 441–454 (2005)
12. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (2000)
13. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by local linear embedding. Science 290(5500), 2323–2326 (2000)
14. Agrafiotis, D.K.: Stochastic proximity embedding. Journal of Computational Chemistry 24(10), 1215–1221 (2003)
15. Donoho, D.L., Grimes, C.: Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. Proceedings of the National Academy of Sciences 100(10), 5591–5596 (2003)
16. de Silva, V., Tenebaum, J.B.: Global versus local methods in nonlinear dimensionality reduction. In: Advances in Neural Information Processing System, pp. 705–712. MIT Press, Cambridge (2002)
17. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10(5), 1299–1319 (1998)
18. Dijkstra, E.W.: Note on two problems in connection with graphs. Numberische Mathematik 1, 269–271 (1959)
19. Seber, G.A.F.: Multivariate Observation. John Wiley & Sons, Chichester (1984)
20. Chen, Y., Crawford, M.M., Ghosh, J.: Applying nonlinear manifold learning to hyperspectral data for land cover classification. In: IEEE International Geoscience and Remote Sensing Symposium, Seoul, South Korea (2005)
21. Bachmann, C.M., Ainsworth, T.L., Fusina, R.A.: Improved Manifold Coordinate Representations of Large-Scale Hyperspectral Scenes. IEEE Transactions on Geoscience and Remote Sensing 44(10), 2786–2803 (2006)
22. Chen, Y., Crawford, M.M., Ghosh, J.: Improved nonlinear manifold learning for land cover classification via intelligent landmark selection. In: IEEE International Geoscience and Remote Sensing Symposium, Denver, Colorado (2006)
23. Kim, W., Chen, Y., Crawford, M.M., Tilton, J.C., Ghosh, J.: Multiresolution manifold learning for classification of hyperspectral data. In: IEEE International Geoscience and Remote Sensing Symposium, pp. 3785–3788 (2007)
24. Kim, W., Crawford, M.M., Ghosh, J.: Spatially adapted manifold learning for classification of hyperspectral imagery with insufficient labeled data. In: IEEE International Geoscience and Remote Sensing Symposium, vol. 1, pp. I-213–I-216 (2008)

When Semi-supervised Learning Meets Ensemble Learning

Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, China
zhouzh@lamda.nju.edu.cn

Abstract. Semi-supervised learning and ensemble learning are two important learning paradigms. The former attempts to achieve strong generalization by exploiting unlabeled data; the latter attempts to achieve strong generalization by using multiple learners. In this paper we advocate generating stronger learning systems by leveraging unlabeled data and classifier combination.

1 Introduction

In many real applications it is difficult to get a large amount of labeled training examples although there may exist abundant unlabeled data, since labeling the unlabeled instances requires human effort and expertise. Exploiting unlabeled data to help improve the learning performance has become a very hot topic during the past decade. There are three major techniques for this purpose [28], i.e., *semi-supervised learning*, *transductive learning* and *active learning*.

Semi-supervised learning [6,36] deals with methods for exploiting unlabeled data in addition to labeled data automatically to improve learning performance, where no human intervention is assumed. Transductive learning [25] also tries to exploit unlabeled data automatically, but it assumes that the unlabeled examples are exactly the test examples. Active learning deals with methods which assume that the learner has some control over the input space, and the goal is to minimize the number of queries from human experts on ground-truth labels for building a strong learner [22]. In this paper we will focus on semi-supervised learning.

From the perspective of generating strong learning systems, it is interesting to see that semi-supervised learning and *ensemble learning* are two important paradigms that were developed almost in parallel and with different philosophies. Semi-supervised learning tries to achieve strong generalization by exploiting unlabeled data, while ensemble learning tries to achieve strong generalization by using multiple learners. From the view of semi-supervised learning, it seems that using unlabeled data to boost the learning performance can be good enough, and so there is no need to involve multiple learners; while from the view of ensemble learning, it seems that using multiple learners can do all the things and therefore there is no need to consider unlabeled data. This partially explains why the MCS community has not paid sufficient attention to semi-supervised ensemble methods [20]. Some successful studies have been reported [3,7,14,15,24,32], while most are semi-supervised boosting methods [3,7,15,24].

In this article we advocate combining the advantages of semi-supervised learning and ensemble learning. Using *disagreement-based semi-supervised learning* [34] as an example, we will discuss why it is good to leverage unlabeled data and classifier combination. After a brief introduction to disagreement-based methods in Section 2, we will discuss on why classifier combination can be helpful to semi-supervised learning in Section 3, discuss on why unlabeled data can be helpful to ensemble learning in Section 4, and finally conclude in Section 5.

2 Disagreement-Based Semi-supervised Learning

Research on disagreement-based semi-supervised learning started from Blum and Mitchell's seminal work on co-training [5]. They considered the situation where data have two *sufficient and redundant* views (i.e., two attribute sets each of which contains sufficient information for constructing a strong learner and is conditionally independent to the other attribute set given the class label). The algorithm trains a learner from each view using the original labeled data. Each learner selects and labels some high-confident unlabeled examples for its peer. Then, each learner is refined using the newly labeled examples provided by its peer. The whole process repeats until no learner changes or a pre-set number of learning rounds is executed.

Blum and Mitchell [5] analyzed the effectiveness of co-training and disclosed that if the two views are conditionally independent, the predictive accuracy of an initial weak learner can be boosted to arbitrarily high using unlabeled data by employing the co-training algorithm. Dasgupta et al. [8] showed that when the two views are sufficient and conditionally independent, the generalization error of co-training is upper-bounded by the disagreement between the two classifiers. Later, Balcan et al. [2] indicated that if a PAC learner can be obtained on each view, the conditional independence assumption or even the weak independent assumption [1] is unnecessary, and a weaker assumption of "expansion" of the underlying data distribution is sufficient for iterative co-training to succeed.

Zhou et al. [35] showed that when there are two sufficient and redundant views, a single labeled training example is able to launch a successful co-training. Indeed, the existence of two sufficient and redundant views is a very luxury requirement. In most real-world tasks this condition does not hold since there is generally only a single attribute set. Thus, the applicability of the standard co-training is limited though Nigam and Ghani [18] showed that if there exist a lot of redundant attributes, co-training can be enabled through view split.

To deal with single view data, Goldman and Zhou [9] proposed a method which trains two learners by using different learning algorithms. The method requires each classifier be able to partition the instance space into equivalence classes, and uses cross validation to estimate the confidences of the two learners as well as the equivalence classes. Zhou and Li [32] proposed the *tri-training* method, which requires neither two views nor special learning algorithms. This method uses three learners and avoids estimating the predictive confidence explicitly. It employs "majority teach minority" strategy in the semi-supervised learning process, that is, if two learners agree on an unlabeled instance but the third learner disagrees, the two learners will label this instance for the third learner. Moreover, classifier combination is exploited to improve generalization. Later, Li and Zhou [14] proposed the *co-forest* method by extending tri-training to include more learners. In co-forest, each learner is improved with unlabeled instances labeled by the ensemble consists of all the other learners, and the final prediction is made by the ensemble of all learners. Zhou and Li [31,33] proposed the first semi-supervised regression algorithm COREG which employs two kNN regressors facilitated with different distance metrics. This algorithm does not require two views either. Later it was extended to a semi-supervised ensemble method for time series prediction with missing data [17].

Previous theoretical studies [2,5,8] worked with two views, and could not explain why these single-view methods can work. Wang and Zhou [26] presented a theoretical analysis which discloses that the key for disagreement-based approaches to succeed is the existence of a large diversity between the learners, and it is unimportant whether the diversity is achieved by using two views, or two learning algorithms, or from other channels.

Disagreement-based semi-supervised learning approaches have been applied to many real-world tasks, such as natural language processing [10,19,21,23], image retrieval [28,29,30], document retrieval [13], spam detection [16], email answering [11], mammogram microcalcification detection [14], etc. In particular, a very effective method which combines disagreement-based semi-supervised learning with active learning for content-based image retrieval has been developed [29,30], and its theoretical analysis was presented recently [27].

3 The Helpfulness of Classifier Combination to Semi-supervised Learning

Here we briefly introduce some of our theoretical results on the helpfulness of classifier combination to semi-supervised learning. Details can be found in a longer version of [26].

Let $\mathcal{H}$ denote a finite hypothesis space and $\mathcal{D}$ the data distribution generated by the ground-truth hypothesis $h^* \in \mathcal{H}$. Let $d(h^i, h^*) = \Pr_{x \in \mathcal{D}}[h^i(x) \neq h^*(x)]$ denote the difference between two classifiers h^i and h^*. Let h_1^i and h_2^i denote the two classifiers in the i-th round, respectively. We consider the following disagreement-based semi-supervised learning process:

Process. *First, we train two initial learners h_1^0 and h_2^0 using the labeled data set $\mathcal{L}$ which contains l labeled examples. Then, h_1^0 selects u number of unlabeled instances from the unlabeled data set $\mathcal{U}$ to label, and puts these newly labeled examples into the data set σ_2 which contains copies of all examples in $\mathcal{L}$; while h_2^0 selects u number of unlabeled instances from $\mathcal{U}$ to label and puts these newly labeled examples into the data set σ_1 which contains copies of all examples in $\mathcal{L}$.*

h_1^1 and h_2^1 are then trained from σ_1 and σ_2, respectively. After that, h_1^1 selects u number of unlabeled instances from $\mathcal{U}$ to label, and updates σ_2 with these newly labeled examples; while h_2^1 selects u number of unlabeled instances to from $\mathcal{U}$ label, and updates σ_1 with these newly labeled examples. The process is repeated for a pre-set number of learning rounds.

We can prove that even when the individual learners could not improve the performance any more, classifier combination is still possible to improve generalization further by using more unlabeled data.

Lemma 1. *Given the initial labeled data set $\mathcal{L}$ which is clean, and assuming that the size of $\mathcal{L}$ is sufficient to learn two classifiers h_1^0 and h_2^0 whose upper bound of the generalization error is $a_0 < 0.5$ and $b_0 < 0.5$ with high probability (more than $1 - \delta$) in the PAC model, respectively, i.e., $l \geq \max[\frac{1}{a_0} \ln \frac{|\mathcal{H}|}{\delta}, \frac{1}{b_0} \ln \frac{|\mathcal{H}|}{\delta}]$. Then h_1^0 selects u number of unlabeled instances from $\mathcal{U}$ to label and puts them into σ_2 which contains all the examples in $\mathcal{L}$, and then h_2^1 is trained from σ_2 by minimizing the empirical risk. If $lb_0 \leq e\sqrt[M]{M!} - M$, then*

$$\Pr[d(h_2^1, h^*) \geq b_1] \leq \delta \ , \tag{1}$$

where $M = ua_0$ and $b_1 = \max[\frac{lb_0 + ua_0 - ud(h_1^0, h_2^1)}{l}, 0]$.

Lemma 1 suggests that the individual classifier h_2^1 can be improved using unlabeled data when $d(h_1^0, h_2^1)$ is larger than a_0.

Considering a simple classifier combination strategy, that is, when two classifiers disagree on a test instance, the classifier which has a higher confidence is relied on. Let h_{com}^i denote the combination of h_1^i and h_2^i, S^i denote the set of examples on which $h_1^i(x) \neq h_2^i(x)$, and $\gamma = Pr_{x \in S^i}[h_{com}^i(x) \neq h^*(x)]$.

Lemma 2. *If $d(h_1^1, h_2^1) > \frac{ua_0 + ub_0 + \left(l(1-2\gamma) - u\right) d(h_1^0, h_2^0)}{u + l(1-2\gamma)}$ and $l < u < c^*$, then*

$$Pr[h_{com}^1(x) \neq h^*(x)] < Pr[h_{com}^0(x) \neq h^*(x)]. \tag{2}$$

Lemma 2 suggests that the classifier combination h_{com}^0 can be improved using unlabeled data when $d(h_1^1, h_2^1)$ is larger than $\frac{ua_0 + ub_0 + \left(l(1-2\gamma) - u\right) d(h_1^0, h_2^0)}{u + l(1-2\gamma)}$. By Lemmas 1 and 2, we have the following theorem.

Theorem 1. *When $d(h_1^0, h_2^0) > a_0 > b_0$ and $\gamma \geq \frac{1}{2} + \frac{u\left(a_0 + b_0 - d(h_1^0, h_2^0)\right)}{2ld(h_1^0, h_2^0)}$, even when $Pr[h_j^1(x) \neq h^*(x)] \geq Pr[h_j^0(x) \neq h^*(x)]$ ($j = 1, 2$), $Pr[h_{com}^1(x) \neq h^*(x)]$ is still less than $Pr[h_{com}^0(x) \neq h^*(x)]$.*

Moreover, we can prove Theorem 2, which suggests that the classifier combination is possible to reach a good performance earlier than the individual classifiers.

Theorem 2. *Suppose $a_0 > b_0$, when $\gamma < \frac{d(h_1^0, h_2^0) + b_0 - a_0}{2d(h_1^0, h_2^0)}$, $Pr[h_{com}^0(x) \neq h^*(x)] < \min[a_0, b_0]$.*

4 The Helpfulness of Unlabeled Data to Ensemble Learning

When there are very few labeled training examples, the necessity of exploiting unlabeled data is obvious, since it is impossible to build a strong ensemble otherwise. So, in this section we will only focus on situation where there are a lot of labeled training examples.

It is well-known that to construct a good ensemble, the base classifiers should be accurate and diverse; however, the diversity is difficult to measure and control [12]. We claim that when there are lots of labeled training examples, unlabeled instances are still helpful since they can help to increase the diversity among the base learners. We will briefly introduce a preliminary study below.

Let $\mathcal{X} = \mathcal{R}^d$ denote the d-dimensional input space and $\mathcal{Y} = \{-1, +1\}$ denote the binary label space. Given labeled training set $\mathcal{L} = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_l, y_l)\}$ and unlabeled training set $\mathcal{U} = \{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n\}$, where $\boldsymbol{x}_i \in \mathcal{X}$, $\boldsymbol{u}_j \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, let $\tilde{\mathcal{L}} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_l\}$ denote the set of unlabeled instances derived from $\mathcal{L}$. Assume that the classifier ensemble $\mathcal{E}$ consists of m linear classifiers $\{\boldsymbol{w}_1 \cdots, \boldsymbol{w}_m\}$, where $\boldsymbol{w}_k \in \mathcal{R}^d$ $(k = 1, \cdots, m)$ is the weight vector of the k-th classifier. Let $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m]$ be the matrix formed by concatenating all weight vectors. Then, we can generate an ensemble by minimizing the loss function

$$V(\mathcal{L},\mathcal{U},\boldsymbol{W}) = \frac{1}{2}\sum_{k=1}^{m} ||\boldsymbol{w}_k||_2^2 + C_1 \cdot V_{acc}(\mathcal{L},\boldsymbol{W}) + C_2 \cdot V_{div}(\mathcal{D},\boldsymbol{W}) \ , \qquad (3)$$

where the first term controls the model complexity, the second term corresponds to the loss of the ensemble in terms of *accuracy* on $\mathcal{L}$ (balanced by C_1), while the third term corresponds to the loss of the ensemble in terms of *diversity* on data set $\mathcal{D}$ (balanced by C_2). Here, we consider two ways to specify $\mathcal{D}$: (1) $\mathcal{D} = \tilde{\mathcal{L}}$, and (2) $\mathcal{D} = \tilde{\mathcal{L}} \bigcup \mathcal{U}$. The first way leads to the method LCD which does not consider unlabeled data, while the second way leads to the method LCDUD which considers both labeled and unlabeled data.

The second loss term in Eq. 3 can be calculated according to

$$V_{acc}(\mathcal{L},\boldsymbol{W}) = \sum_{k=1}^{m}\sum_{i=1}^{l} loss(\boldsymbol{w}_k, \boldsymbol{x}_i, y_i) \ , \qquad (4)$$

where $loss(\boldsymbol{w}_k, \boldsymbol{x}_i, y_i)$ measures the loss of the k-th base classifier, i.e., $\boldsymbol{w}_k$, on the i-th labeled training example, i.e., $(\boldsymbol{x}_i, y_i)$. Here we calculate it using the l_2 norm

$$loss(\boldsymbol{w}_k, \boldsymbol{x}_i, y_i) = \begin{cases} 0 & \text{if } y_i\langle \boldsymbol{w}_k, \boldsymbol{x}_i\rangle \geq 1 \\ (1 - y_i\langle \boldsymbol{w}_k, \boldsymbol{x}_i\rangle)^2 & \text{if } y_i\langle \boldsymbol{w}_k, \boldsymbol{x}_i\rangle < 1 \end{cases}$$

where $\langle\cdot,\cdot\rangle$ denotes the dot product between vectors.

We calculate the third term in Eq. 3 by considering the *prediction difference* between each pair of base classifiers, i.e.,

$$V_{div}(\mathcal{D}, \boldsymbol{W}) = \sum_{p=1}^{m-1} \sum_{q=p+1}^{m} d(\boldsymbol{w}_p, \boldsymbol{w}_q, \mathcal{D}) \ , \tag{5}$$

where

$$d(\boldsymbol{w}_p, \boldsymbol{w}_q, \mathcal{D}) = \begin{cases} 0 & \text{if } \mathcal{D} = \emptyset \\ \frac{\sum_{\boldsymbol{x} \in \mathcal{D}} \mathbf{sign}(\langle \boldsymbol{w}_p, \boldsymbol{x} \rangle) \cdot \mathbf{sign}(\langle \boldsymbol{w}_q, \boldsymbol{x} \rangle)}{|\mathcal{D}|} & \text{if } \mathcal{D} \neq \emptyset \end{cases} .$$

By putting Eqs. 4 and 5 into Eq. 3, and approximating $\mathbf{sign}(\cdot)$ by $\mathbf{tanh}(\cdot)$, the resulting loss function turns to be a continuous and differentiable function of the model parameters $\boldsymbol{W}$. Thus our goal becomes to find the optimal model $\boldsymbol{W}^*$ which minimizes

$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} V(\mathcal{L}, \mathcal{U}, \boldsymbol{W}) \ . \tag{6}$$

We initialize $\boldsymbol{W}$ by generating each classifier $\boldsymbol{w}_k$ from a bootstrap sample of $\mathcal{L}$, i.e., $\mathcal{L}_k = \{(\boldsymbol{x}_1^k, y_1^k), \cdots, (\boldsymbol{x}_l^k, y_l^k)\}$, by solving the SVM-style optimization problem

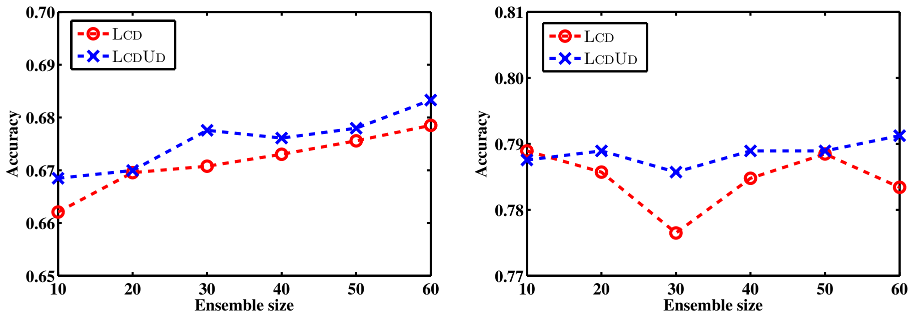
$$\min_{\boldsymbol{w}_k, \boldsymbol{\xi}} \frac{1}{2} ||\boldsymbol{w}_k||_2^2 + C \sum_{i=1}^{l} \xi_i^k \quad \text{s.t. } \ y_i^k \langle \boldsymbol{w}_k, \boldsymbol{x}_i^k \rangle \geq 1 - \xi_i^k, \ \xi_i^k \geq 0 \ .$$

where $\boldsymbol{\xi} = [\xi_1^k, \xi_2^k, \cdots, \xi_l^k]$. The above problem falls into the category of quadratic programming (QP) and can be solved efficiently by a number of methods off-the-shelf. Then, we solve Eq. 6 by gradient descent.

Figure 1 shows some preliminary results on data sets `g241n`[1] and `vehicle` [4]. For each data set, a half data is randomly chosen to form the test set. Among the remaining data, 5% are used as labeled training examples while 95% are used as unlabeled instances. The experiments are repeated for ten times with random data splits. The parameters C_1 and C_2 are both set to 1. In Figure 1 the horizontal axis in each subfigure shows the size of the ensembles (from 10 to 60 with an interval of 10), and the vertical axis shows the average accuracy. The results show that LcdUd can outperform Lcd, while the only difference between LcdUd and Lcd is that the former considers the usefulness of unlabeled data.

It is worth noting that the above method is far from an excellent one since it does not distinguish the priorities of the contribution from labeled data and unlabeled data. Ideally, the accuracy and diversity on labeled data should be considered at first to form a pool of comparable ensembles, and then from the pool an ensemble with high diversity on unlabeled data is selected. Powerful ensemble methods would be developed along this direction.

[1] http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html



(a) `g241n` ($N = 1500,\ d = 241$) (b) `vehicle` ($N = 435,\ d = 26$)

Fig. 1. Comparing the performance of LCD and LCDUD. N is the number of instances; d is the dimensionality.

5 Conclusion

Semi-supervised learning and ensemble learning are two well-developed paradigms for improving generalization. Although there are some studies of semi-supervised ensemble methods, the MCS community has not devoted much effort to this line of research. In this article we argue that

– Classifier combination is helpful to semi-supervised learning. There are at least two reasons: 1) the performance of classifier combination can be improved further even though the individual learners could not be improved using unlabeled data; 2) the classifier combination can reach a good performance earlier than individual learners.
– Unlabeled data are helpful to ensemble learning. There are at least two reasons: 1) when there are very few labeled training examples, unlabeled data have to be exploited for constructing a strong ensemble; 2) unlabeled data can be used to help increase the diversity of base learners.

Our arguments were made on disagreement-based semi-supervised learning approaches, however, they are possible to generalize to other kinds of semi-supervised learning and ensemble learning approaches. We believe that semi-supervised ensemble methods are very worth studying. Moreover, we think it is possible to derive effective diversity controls for ensemble learning by considering the usefulness of unlabeled data.

Acknowledgments

The author wants to thank Wei Wang and Min-Ling Zhang for their help. This research was supported by the National Science Foundation of China (60635030, 60721002), the National High Technology Research and Development Program of China (2007AA01Z169), the Jiangsu Science Foundation (BK2008018) and the Jiangsu 333 High-Level Talent Cultivation Program.

References

1. Abney, S.: Bootstrapping. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, pp. 360–367 (2002)
2. Balcan, M.-F., Blum, A., Yang, K.: Co-training and expansion: Towards bridging theory and practice. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 17, pp. 89–96. MIT Press, Cambridge (2005)
3. Bennett, K., Demiriz, A., Maclin, R.: Exploiting unlabeled data in ensemble methods. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, pp. 289–296 (2002)
4. Blake, C., Keogh, E., Merz, C.J.: UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine, CA (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
5. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the 11th Annual Conference on Computational Learning Theory, Madison, WI, pp. 92–100 (1998)
6. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press, Cambridge (2006)
7. d'Alché-Buc, F., Grandvalet, Y., Ambroise, C.: Semi-supervised MarginBoost. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems, vol. 14, pp. 553–560. MIT Press, Cambridge (2002)
8. Dasgupta, S., Littman, M., McAllester, D.: PAC generalization bounds for co-training. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems, vol. 14, pp. 375–382. MIT Press, Cambridge (2002)
9. Goldman, S., Zhou, Y.: Enhancing supervised learning with unlabeled data. In: Proceedings of the 17th International Conference on Machine Learning, San Francisco, CA, pp. 327–334 (2000)
10. Hwa, R., Osborne, M., Sarkar, A., Steedman, M.: Corrected co-training for statistical parsers. In: Working Notes of the ICML 2003 Workshop on the Continum from Labeled to Unlabeled Data in Machine Learning and Data Mining, Washington, DC (2003)
11. Kockelkorn, M., Lüneburg, A., Scheffer, T.: Using transduction and multi-view learning to answer emails. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS, vol. 2838, pp. 266–277. Springer, Heidelberg (2003)
12. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning 51(2), 181–207 (2003)
13. Li, M., Li, H., Zhou, Z.-H.: Semi-supervised document retrieval. Information Processing & Management 45(3), 341–355 (2009)
14. Li, M., Zhou, Z.-H.: Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans 37(6), 1088–1098 (2007)
15. Mallapragada, P.K., Jin, R., Jain, A.K., Liu, Y.: SemiBoost: Boosting for semi-supervised learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (2009)
16. Mavroeidis, D., Chaidos, K., Pirillos, S., Christopoulos, D., Vazirgiannis, M.: Using tri-training and support vector machines for addressing the ECML-PKDD 2006 Discovery Challenge. In: Proceedings of ECML-PKDD 2006 Discovery Challenge Workshop, Berlin, Germany, pp. 39–47 (2006)

17. Mohamed, T.A., El Gayar, N., Atiya, A.F.: A co-training approach for time series prediction with missing data. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 93–102. Springer, Heidelberg (2007)
18. Nigam, K., Ghani, R.: Analyzing the effectiveness and applicability of co-training. In: Proceedings of the 9th ACM International Conference on Information and Knowledge Management, Washington, DC, pp. 86–93 (2000)
19. Pierce, D., Cardie, C.: Limitations of co-training for natural language learning from large data sets. In: Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing, Pittsburgh, PA, pp. 1–9 (2001)
20. Roli, F.: Semi-supervised multiple classifier systems: Background and research directions. In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS 2005. LNCS, vol. 3541, pp. 1–11. Springer, Heidelberg (2005)
21. Sarkar, A.: Applying co-training methods to statistical parsing. In: Proceedings of the 2nd Annual Meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, PA, pp. 95–102 (2001)
22. Settles, B.: Active learning literature survey. Technical Report 1648, Department of Computer Sciences, University of Wisconsin at Madison, Wisconsin, WI (2009), http://pages.cs.wisc.edu/~bsettles/pub/settles.activelearning.pdf
23. Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlen, P., Baker, S., Crim, J.: Bootstrapping statistical parsers from small data sets. In: Proceedings of the 11th Conference on the European Chapter of the Association for Computational Linguistics, Budapest, Hungary, pp. 331–338 (2003)
24. Valizadegan, H., Jin, R., Jain, A.K.: Semi-supervised Boosting for multi-class classification. In: Proceedings of the 19th European Conference on Machine Learning, Antwerp, Belgium, pp. 522–537 (2008)
25. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
26. Wang, W., Zhou, Z.-H.: Analyzing co-training style algorithms. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS, vol. 4701, pp. 454–465. Springer, Heidelberg (2007)
27. Wang, W., Zhou, Z.-H.: On multi-view active learning and the combination with semi-supervised learning. In: Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, pp. 1152–1159 (2008)
28. Zhou, Z.-H.: Learning with unlabeled data and its application to image retrieval. In: Yang, Q., Webb, G. (eds.) PRICAI 2006. LNCS, vol. 4099, pp. 5–10. Springer, Heidelberg (2006)
29. Zhou, Z.-H., Chen, K.-J., Dai, H.-B.: Enhancing relevance feedback in image retrieval using unlabeled data. ACM Transactions on Information Systems 24(2), 219–244 (2006)
30. Zhou, Z.-H., Chen, K.-J., Jiang, Y.: Exploiting unlabeled data in content-based image retrieval. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS, vol. 3201, pp. 525–536. Springer, Heidelberg (2004)
31. Zhou, Z.-H., Li, M.: Semi-supervised regression with co-training. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, pp. 908–913 (2005)
32. Zhou, Z.-H., Li, M.: Tri-training: Exploiting unlabeled data using three classifiers. IEEE Transactions on Knowledge and Data Engineering 17(11), 1529–1541 (2005)
33. Zhou, Z.-H., Li, M.: Semi-supervised regression with co-training style algorithms. IEEE Transactions on Knowledge and Data Engineering 19(11), 1479–1493 (2007)
34. Zhou, Z.-H., Li, M.: Semi-supervised learning by disagreement. In: Knowledge and Information Systems (in press)

35. Zhou, Z.-H., Zhan, D.-C., Yang, Q.: Semi-supervised learning with very few labeled training examples. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, Vancouver, Canada, pp. 675–680 (2007)
36. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI(2006), http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf

Author Index